

jc560 U.S. PTO

09/003972

APPENDIX B

Inventor: Everett W. Stouff

Title: System for Converting Scrolling
Display To Non-Scrolling Columnar
Display

BEST AVAILABLE COPY

```

1  /**
2   *  %W% %E% Everett Stoub
3   *
4   *  Copyright (c) '96-'97 ICON Web Products, Inc. All Rights Reserved.
5   *
6   *  ez_html applet displays snaking text with in-line graphics
7   *  and produces a reader screen frame when clicked
8   *
9   *
10  *  applet param specifications
11  *
12  *      fileName    value=name_of_file    (required)
13  *                  extension must be .htm or .html for html parsing
14  *                  e.g. <param name="fileName" value="ezsample.htm">
15  *
16  *      maxColumns  value=an_integer      (optional)
17  *                  specifies upper limit of column number in APPLET display only
18  *                  e.g. <param name="maxColumns" value="2">
19  *
20  *      frameType   value=special_name    (optional)
21  *                  e.g. <param name="frameType" value="LINE">          default frameSize = 1
22  *                  e.g. <param name="frameType" value="FLAT">         default frameSize = 1
23  *                  e.g. <param name="frameType" value="RAISED">       default frameSize = 2
24  *                  e.g. <param name="frameType" value="DROPPED">      default frameSize = 2
25  *                  e.g. <param name="frameType" value="ETCHED">       default frameSize = 4
26  *
27  *      frameSize   value=an_integer      (optional)
28  *                  specifies width of optional frame
29  *                  e.g. <param name="frameSize" value="2">
30  *
31  *
32  *  HTML tag parsing honors the following commands and attributes
33  *
34  *      • <title>...</title>          title text delimiters
35  *
36  *      • <base>                      document relationships
37  *        href                        relative or absolute url for hyperlink
38  *
39  *      • <body>...</body>           body text delimiters
40  *        bgcolor=color               background color (#hhhhh or name)
41  *        text=color                  default text color
42  *        link=color                  hypertext link color
43  *        background=URL              tiled applet & reader background
44  *
45  *      • <table>...</table>         table tag & range (replaced with hot image)
46  *
47  *
48  *  Gauthor Everett Stoub %I%, %G%
49  */
50  import java.awt.*;
51  import java.awt.image.*;
52  import java.applet.*;
53  import java.net.*;
54  import java.io.*;
55  import java.util.*;
56
57  public class ez_html extends Applet
58  {
59      public static Color    backColor    = new Color(239,239,239);
60      public static Color    foreColor    = Color.black;
61      public static Color    linkColor    = Color.blue;
62
63      public static int      maxColumns    = 15;
64
65      public _f_rs_db        frame        = null;
66      public String          title        = null;
67      private String         article      = null;
68      private _p_rs_db       titlePanel   = null;
69      private _p_rs_db       bodyPanel    = null;
70      private _o_im_db       imageDB      = null;

```

```
1      private _o_tb_db      tableDB      = null;
2      private _o_fl_db      formsDB      = null;
3
4      private Image          backImage     = null;
5      private URL            baseURL       = null;
6
7      private MediaTracker   tracker       = null;
8
9      private final int      NONE          = 0;
10     private final int      LINE          = 1;
11     private final int      FLAT          = 2;
12     private final int      RAISED       = 3;
13     private final int      DROPPED      = 4;
14     private final int      ETCHED       = 5;
15
16     private int             frameType     = NONE;
17     private int             frameSize     = 0;
18     private Insets          insets       = new Insets(0, 0, 0, 0);
19     public final static String CS        = "Release to create reader
20 screen...";
21     public final static String RS        = "Release to view reader screen...";
22     public final static String WS        = "Wait for new reader screen...";
23     public final static String ZM        = "Click to read...";
24
25     private static final void dbg(String s)
26     {
27         if (debug) System.out.println(s);
28     }
29     private static boolean debug = false;    // Print debugging info?
30
31     public void disposeFrame(_f_rs_db disposee)
32     {
33         if (disposee != null)
34         {
35             Event evt = new Event(disposee, Event.WINDOW_DESTROY, null);
36             Component co[] = disposee.getComponents();
37             for (int i = 0; i < co.length; i++)
38                 co[i].handleEvent(evt);
39             disposee.dispose();
40         }
41         if (disposee == frame)
42         {
43             frame = null;
44         }
45     }
46
47     public Image getBackImage()
48     {
49         return this.backImage;
50     }
51
52     public URL getDocumentBase()
53     {
54         if (baseURL != null)
55             return baseURL;
56         else
57             return super.getDocumentBase();
58     }
59
60     public _o_im_db getImageDB()
61     {
62         if (this.imageDB == null)
63             this.imageDB = new _o_im_db(this);
64
65         return this.imageDB;
66     }
67
68     public _o_tb_db getTableDB()
69     {
70         if (this.tableDB == null)
```

```

1      this.tableDB = new _o_tb_db(this);
2
3      return this.tableDB;
4  }
5
6  public _o_fl_db getFormDB()
7  {
8      if (this.formsDB == null)
9          this.formsDB = new _o_fl_db(this);
10
11     return this.formsDB;
12 }
13
14 public Color getBackColor()
15 {
16     return this.backColor;
17 }
18
19 public Color getForeColor()
20 {
21     return this.foreColor;
22 }
23
24 public String getTitle()
25 {
26     return this.title;
27 }
28
29 public MediaTracker getTracker()
30 {
31     if (this.tracker == null)
32         this.tracker = new MediaTracker(this);
33
34     return this.tracker;
35 }
36
37 public void init()
38 {
39     String s = getParameter("fileName");
40     if (s != null)
41     {
42         article = _o_ht_db.getArticle(s, this);
43     }
44
45     String m = getParameter("maxColumns");
46     if (m != null)
47     {
48         maxColumns = Math.max
49             (
50                 _o_tg_db.parseInt(m, maxColumns)
51                 , 1
52             );
53     }
54
55     this.setLayout(new BorderLayout());
56
57     if (title != null)
58     {
59         // dbg("titlePanel: "+title);
60
61         String t = getParameter("showTitle");
62         if (t != null)
63         {
64             if (t.toLowerCase().equals("yes"))
65             {
66                 titlePanel = new _p_rs_db
67                     (
68                         "<font color=\"#0000b3\"><h2 align=center>"+title
69                         , this
70                         , null
71                         , false
72                         , 1
73                     )
74                 // the browser is the frame
75                 // NOT the reader screen
76                 // fixed to center the title

```



```

1      , false          // no paging indicator marks
2      , false          // no page position footer
3      , false          // not zoomable
4      );
5      titlePanel.textAlign = _p_rs_db.CENTER;
6      this.add("North", titlePanel);
7      }
8      }
9      }
10
11     bodyPanel = new _p_rs_db
12     ( article
13     , this
14     , null              // the browser is the frame
15     , false            // NOT the reader screen
16     , maxColumns        // completely automatic
17     , true              // use paging indicator marks
18     , false            // no page position footer
19     , true              // zoomable
20     );
21     this.add("Center", bodyPanel);
22
23     String b = getParameter("frameType");
24     if (b != null)
25     {
26         b = b.toUpperCase();
27         if (b.equals("LINE")) { frameSize = 1; frameType = LINE; }
28         if (b.equals("FLAT")) { frameSize = 1; frameType = FLAT; }
29         if (b.equals("RAISED")) { frameSize = 2; frameType = RAISED; }
30         if (b.equals("DROPPED")) { frameSize = 2; frameType = DROPPED; }
31         if (b.equals("ETCHED")) { frameSize = 4; frameType = ETCHED; }
32     }
33
34     String w = getParameter("frameSize");
35     if (w != null) try
36     {
37         frameSize = Math.max
38         ( Integer.parseInt(w)
39         , 1
40         );
41     }
42     catch (NumberFormatException nf) {}
43
44     switch (frameType)
45     {
46         case LINE:
47             insets.bottom = frameSize; break;
48         case FLAT:
49         case RAISED:
50         case DROPPED:
51         case ETCHED:
52             insets.top = frameSize;
53             insets.left = frameSize;
54             insets.bottom = frameSize;
55             insets.right = frameSize;
56     }
57 }
58
59 public Insets insets()
60 {
61     return insets;
62 }
63
64 public boolean mouseDown(Event evt, int x, int y)
65 {
66     // dbg("ez_html.mouseDown()");
67
68     if (frame != null)
69     {
70         showStatus(RS);

```

```
1      }
2      else
3      {
4          showStatus(CS);
5      }
6
7      return true;
8  }
9
10     public boolean mouseEnter(Event evt, int x, int y)
11     {
12         showStatus(ZM);
13
14         return true;
15     }
16
17     public boolean mouseExit(Event evt, int x, int y)
18     {
19         showStatus(_p_rs_db.MT);
20
21         return true;
22     }
23
24     public boolean mouseMove(Event evt, int x, int y)
25     {
26         showStatus(ZM);
27
28         return true;
29     }
30
31     public boolean mouseUp(Event evt, int x, int y)
32     {
33         // dbg("ez_html.mouseUp()");
34
35         if (frame != null)
36         {
37             // dbg("ez_html.mouseUp(): restoring _f_rs_db");
38             frame.toFront();
39         }
40         else
41         {
42             // dbg("ez_html.mouseUp(): making new _f_rs_db");
43             showStatus(WS);
44             frame = new _f_rs_db
45                 (
46                 title
47                 , article
48                 , this
49                 , 15
50                 , true
51                 , true
52                 );
53             // max columns
54             // present markers
55             // present footer
56
57             return true;
58         }
59     }
60
61     public final void paint(Graphics g)
62     {
63         // dbg("applet.paint()");
64
65         int i;
66         Dimension d = size();
67         if (g != null) switch (frameType)
68         {
69             case LINE:
70                 {
71                     g.setColor(getForeground());
72                     for (i = 0; i < frameSize; i++)
73                         g.drawLine(0,d.height-i-1,d.width-i-1,d.height-i-1);
74                     break;
75                 }
76             case FLAT:
77                 {
78                     g.setColor(getForeground());
```

```

1         for (i = 0; i < frameSize; i++)
2             g.drawRect(i,i,d.width-2*i-1,d.height-2*i-1);
3     } break;
4     case RAISED:
5     case DROPPED:
6     case ETCHED:
7     { boolean raise = frameType==RAISED;
8       int R = (getForeground().getRed()+ getBackground().getRed() )/2;
9       int G = (getForeground().getGreen()+getBackground().getGreen())/2;
10      int B = (getForeground().getBlue()+ getBackground().getBlue() )/2;
11      g.setColor(new Color(R,G,B));
12      for (i = 0; i < frameSize; i++)
13      {
14          if (frameType == ETCHED) raise = i >= frameSize/2;
15          g.draw3DRect(i,i,d.width-2*i-1,d.height-2*i-1,raise);
16      }
17      if (frameType == ETCHED)
18      {
19          g.setColor(getForeground());
20          g.drawLine(0,0,frameSize/2,frameSize/2);
21          g.drawLine(d.width-frameSize,d.height-frameSize,d.width-
22 frameSize/2,d.height-frameSize/2);
23          g.setColor(getBackground());
24          g.drawLine(frameSize/2,frameSize/2,frameSize,frameSize);
25          g.drawLine(d.width-frameSize/2,d.height-frameSize/2,d.width-
26 1,d.height-1);
27      }
28      else
29      {
30          g.setColor((frameType==RAISED)?getBackground():getForeground());
31          g.drawLine(0,0,frameSize,frameSize);
32          g.setColor((frameType==RAISED)?getForeground():getBackground());
33          g.drawLine(d.width-frameSize,d.height-frameSize,d.width-
34 1,d.height-1);
35      }
36    }
37  }
38  }
39
40  public void setBackgroundColor(Color color)
41  {
42      this.backColor = color;
43  }
44
45  public void setBackImage(Image image)
46  {
47      this.backImage = image;
48  }
49
50  public void setBaseURL(URL baseURL)
51  {
52      this.baseURL = baseURL;
53  }
54
55  public void setForeColor(Color color)
56  {
57      this.foreColor = color;
58  }
59
60  public void setLinkColor(Color color)
61  {
62      this.linkColor = color;
63  }
64
65  public void setTitle(String title)
66  {
67      this.title = title;
68  }
69
70  public void update(Graphics g)

```

```

1      {
2      // fillRect is NOT performed here to eliminate blank screen boredom during
3      offscreen drawing
4
5      // g.setColor(getBackground());
6      // g.fillRect(0, 0, width, height);
7      // g.setColor(getForeground());
8
9      paint(g);
10     }
11 }
12
13 /**
14  * %W% %E% Everett Stoub
15  *
16  * Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
17  *
18  * an object to read html documents and parse head content
19  *
20  * this class is not downloaded unless needed
21  *
22  * @author Everett Stoub %I%, %G%
23  */
24 import java.awt.*;
25 import java.io.*;
26 import java.net.*;
27 import java.util.*;
28
29 public class _o_ht_db extends Object
30 {
31     private ez_html          ez_HTML      = null;
32
33     private final static String alert_0    = "PLEASE READ - A Java ";
34     private final static String alert_1    = "Exception has occurred. ";
35     private final static String alert_2    = "Most likely, there has been a
36     transmission failure with one or more applet files. A Refresh or Reload command on
37     your browser will sometimes lead to a successful download. ";
38     private final static String alert_3    = "An attempt to read a local file has
39     been detected! In order to read this article, you must download the applet using
40     hyper-text transport protocols (http://...) from a web server. Please try again using
41     http with a valid URL. ";
42     private final static String alert_4    = "If the problem persists, please contact
43     the Web Master.";
44
45     public final static String HL          = " ";
46     public final static String BS          = "<base";
47     public final static String HF          = "href";
48     public final static String MT          = " ";
49     public final static String FR          = "<p>";
50
51     private static final void dbg(String s)
52     {
53         if (debug) System.out.println(s);
54     }
55     private static boolean debug = true;          // Print debugging info?
56
57     public static String getArticle(String fileName, ez_html ez_HTML)
58     {
59         String article = MT;
60
61         if (fileName != null) try
62         URL, etc.
63         {
64             URL url = new URL(ez_HTML.getDocumentBase(), fileName);
65             InputStream is = url.openStream();
66             BufferedInputStream bis = new BufferedInputStream(is);
67
68             int len = 0;
69             byte buffer[] = new byte[512];
70             StringBuffer pageText = new StringBuffer();

```

```

1      while ((len = bis.read(buffer)) > 0)
2          pageText.append(new String(buffer,0,0,len));
3
4      bis.close();
5      is.close();
6
7      boolean isHTML = fileName.toLowerCase().endsWith(".htm")
8          || fileName.toLowerCase().endsWith(".html");    // other
9      indicators would be more helpful...
10
11         if (isHTML)                                     //
12     protect literal sections from stripping
13     {
14         StringBuffer articleBuffer = new StringBuffer();
15         Vector pieces = new Vector(50);                // String
16     segments of pageText distinguished by filtering
17         Vector status = new Vector(50);                //
18     Boolean status of each segment: true of filtering is needed
19
20         String MText = pageText.toString();
21         String LText = MText.toLowerCase();
22
23         // the plaintext tag preempts all subsequent tags
24
25         int plaintext = LText.indexOf("<plaintext>");
26         if (plaintext > -1)                             // rare:
27     a deprecated tag by HTML 3.2
28         {
29             pieces.addElement(MText.substring(0,plaintext));
30             status.addElement(new Boolean(true));        //
31     filtering is needed
32
33             pieces.addElement(lineCanon(MText.substring(plaintext), false));
34             status.addElement(new Boolean(false));        //
35     filtering is complete
36         }
37         else
38         {
39             pieces.addElement(MText);
40             status.addElement(new Boolean(true));        //
41     filtering is needed
42         }
43
44         // <textarea>...</textarea> & <pre>...</pre> ranges are taken literally
45
46         while (status.size() > 0)
47         {
48             if (((Boolean)status.firstElement()).booleanValue())    //
49     filtering needed
50             {
51                 MText = (String)pieces.firstElement();
52                 LText = MText.toLowerCase();
53
54                 String TE = "</textarea>";
55                 String PE = "</pre>";
56                 int began, ended;
57                 if ((ended = LText.indexOf(TE)) > -1
58                     && (began = LText.lastIndexOf("<textarea",ended)) > -1)
59                 {
60                     ended += TE.length();                //
61     inclusive terminus
62
63                     pieces.setElementAt(MText.substring(0,began), /*
64     replace original */ 0);
65                     status.setElementAt(new Boolean(true), /*
66     filtering needed */ 0);
67
68                     pieces.insertElementAt(lineCanon(MText.substring(began,
69     ended), false), 1);

```

```

1      status.insertElementAt(new Boolean(false), /*
2      filtering complete */ 1);
3
4      pieces.insertElementAt(MCtext.substring(ended),
5      2);
6      status.insertElementAt(new Boolean(true), /*
7      filtering needed */ 2);
8      }
9      else
10     if ((ended = LCtext.indexOf(PE)) > -1
11     && (began = LCtext.lastIndexOf("<pre",ended)) > -1)
12     {
13         ended += PE.length(); //
14         inclusive terminus
15
16         pieces.setElementAt(MCtext.substring(0,began), /*
17         replace original */ 0);
18         status.setElementAt(new Boolean(true), /*
19         filtering needed */ 0);
20
21         pieces.insertElementAt(lineCanon(MCtext.substring(began,
22         ended), false), 1);
23         status.insertElementAt(new Boolean(false), /*
24         filtering complete */ 1);
25
26         pieces.insertElementAt(MCtext.substring(ended),
27         2);
28         status.insertElementAt(new Boolean(true), /*
29         filtering needed */ 2);
30     }
31     else // filter
32     it as html & tack it on
33     {
34         articleBuffer.append(lineCanon(MCtext, true)); //
35         filtering complete: tack it on
36
37         pieces.removeElementAt(0); // remove
38         original
39         status.removeElementAt(0); // remove
40         original
41     }
42     }
43     else //
44     filtering is complete: tack it on
45     {
46         articleBuffer.append((String)pieces.firstElement());
47
48         pieces.removeElementAt(0);
49         status.removeElementAt(0);
50     }
51 }
52 article = htmlBodyText(articleBuffer.toString(), ez_HTML);
53 }
54 else
55 {
56     article = htmlBodyText(lineCanon(pageText.toString(), false),
57     ez_HTML);
58 }
59 }
60 catch (MalformedURLException mu)
61 {
62     StringBuffer msg = new StringBuffer();
63     msg.append(alert_0).append("Malformed URL
64     ").append(alert_1).append(alert_2).append(alert_4);
65
66     article = msg.toString();
67
68     String e = "Could not open "+fileName;
69     ez_HTML.showStatus(e);
70     System.out.println(e+": "+mu);

```

```

1      }
2      catch (IOException io)
3      {
4          StringBuffer msg = new StringBuffer();
5
6          msg.append(alert_0).append("Input/Output").append(alert_1).append(alert_2).append(
7          alert_4);
8
9          article = msg.toString();
10
11         String e = "Could not open "+fileName;
12         ez_HTML.showStatus(e);
13         System.out.println(e+": "+io);
14     }
15     catch (SecurityException se)
16     {
17         StringBuffer msg = new StringBuffer();
18         msg.append(alert_0).append("Security
19 ") .append(alert_1).append(alert_3).append(alert_4);
20
21         article = msg.toString();
22
23         String e = "Could not open "+fileName;
24         ez_HTML.showStatus(e);
25         System.out.println(e+": "+se);
26     }
27     else
28     {
29         StringBuffer msg = new StringBuffer();
30         msg.append(alert_0).append("Input/Output").append(alert_1);
31         msg.append("No article file specification was detected! ");
32         msg.append(alert_2).append(alert_4);
33
34         article = msg.toString();
35
36         String e = "No fileName specified";
37         ez_HTML.showStatus(e);
38         System.out.println(e);
39     }
40
41     return article;
42 }
43
44 public static String htmlBodyText(String itsText, ez_html ez_HTML)
45 {
46     String newText = new String(itsText);
47     String LcText = newText.toLowerCase();
48
49     // look for [<!--...-->] html comments and eliminate them
50
51     String CE = "-->";
52
53     int began, ended = LcText.indexOf(CE);
54     while (ended > -1)
55     comment
56     {
57         began = LcText.lastIndexOf("<!--", ended);
58     comment
59         if (began > -1)
60         {
61             ended += CE.length();
62
63             StringBuffer text = new StringBuffer();
64             text.append(newText.substring(0, began));
65             text.append(newText.substring(ended));
66
67             newText = text.toString();
68             LcText = newText.toLowerCase();
69             ended = LcText.indexOf(CE);
70         }

```

```

1         else
2         {
3             ended = -1; // format error exit path
4         }
5     }
6
7     // look for [<title...>title text</title...>]
8
9     began = LCtext.indexOf("<title");
10    if (began > -1)
11    {
12        began = LCtext.indexOf(">", began)+1; // start of title
13        if (began > 0 && began < LCtext.length())
14        {
15            ended = LCtext.indexOf("</title", began);
16            if (ended > -1)
17            {
18                ez_HTML.setTitle(newText.substring(began, ended));
19            }
20        }
21    }
22
23    began = LCtext.indexOf("BS");
24    if (began > -1)
25    {
26        began += BS.length();
27        ended = LCtext.indexOf(">", began)+1; // end of base
28        if (ended > 0 && ended < LCtext.length())
29        {
30            String itsSpec =
31            _o_tg_db.getTagAttribString(newText.substring(began, ended), HF);
32            try
33            {
34                ez_HTML.setBaseURL(new URL(ez_HTML.getDocumentBase(), itsSpec));
35            }
36            catch (MalformedURLException m)
37            {
38                ez_HTML.setBaseURL(null);
39            }
40        }
41    }
42
43    // look for [<form...>table specs</form...>] recursively number document forms &
44    elements
45
46    String fb = _p_rs_db.PF + "form";
47    String fe = _p_rs_db.PF + _p_rs_db.SL + "form" + _p_rs_db.SF;
48
49    int start, formNumber = 0;
50
51    ended = LCtext.indexOf(fe); // the first end-of-form
52    began = LCtext.lastIndexOf(fb, ended); // its (prior) start-of-
53    form
54    start = LCtext.indexOf(_p_rs_db.SF, began); // its start-of-form
55    endpoint
56    while (-1 < began && began < start && start < ended)
57    {
58        formNumber++;
59        start += _p_rs_db.SF.length(); // start-of-form content
60    range start point
61
62        int trail = ended + fe.length(); // beginning of trailing
63    content
64
65        String orig_tag = LCtext.substring(began, start); // just the original form
66    tag
67        _o_fm_db theForm = new _o_fm_db(formNumber, orig_tag);
68        ez_HTML.getFormDB().newForm(theForm);
69
70        String formContent = newText.substring(start, ended);

```



```

1      String newContent = theForm.replaceContent(formContent);
2
3      StringBuffer text = new StringBuffer();
4      text.append(newText.substring(0,began))           // preceding text,
5  excluding original form tag
6      .append(newContent)                             // new form range content
7  (modified elements plus orig html content)
8      .append(newText.substring(trail));               // succeeding content
9  text, excluding original end form tag
10
11      ended -= trail - began;                          // this was dropped
12      ended += newContent.length();                   // this was added
13
14      newText = text.toString();
15      LCtext = newText.toLowerCase();
16      ended = LCtext.indexOf(fe, ended);               // the next end-of-form
17      began = LCtext.lastIndexOf(fb, ended);           // its (prior) start-of-
18  form
19      start = LCtext.indexOf(_p_rs_db.SF, began);       // its start-of-form
20  endpoint
21  }
22
23      // look for [<table...>table specs</table...> recursively: load and replace specs
24
25      String TE = "</table>";
26
27      ended = LCtext.indexOf(TE);
28      int tableNumber = 0;
29      while (ended > -1)                                // the first end-of-table
30      {
31          began = LCtext.lastIndexOf("<table",ended);   // its (prior) start-of-
32  table
33          if (began > -1)
34          {
35              ended += TE.length();                     // inclusive terminus
36              tableNumber++;
37
38              // dbg("! - new table (" + tableNumber + "): range = (" + began + ",
39  "+ ended + ")");
40
41              _p_tb_db.newTable = new _p_tb_db
42              (    newText.substring(began, ended)
43                ,    new Integer(tableNumber)
44                ,    ez_HTML
45                );
46              ez_HTML.getTableDB().addTable(newTable);
47
48              StringBuffer text = new StringBuffer();
49              text.append(newText.substring(0,began))   // preceding text
50              .append(newTable.addCaption(true))         // superior caption, if
51  any
52              .append(newTable.addCaption(false))        // inferior caption, if
53  any, in superior position
54              .append(_p_rs_db.PF)
55              .append(_p_rs_db.TB)                       // private table tag
56              .append(_p_rs_db.BL)
57              .append(_p_rs_db.VL)
58              .append(_p_rs_db.EQ)
59              .append(newTable.getTableNumber())
60              .append(_p_rs_db.SF)                       // substitute text, e.g.
61  <tblt value=1>
62              // .append(newTable.addCaption(false))    // inferior caption, if
63  any; shifted to superior position
64              .append(newText.substring(ended));         // succeeding text
65
66              newText = text.toString();
67              LCtext = newText.toLowerCase();
68              ended = LCtext.indexOf(TE);
69          }
70      else

```

```

1      {
2          ended = -1;                                // format error exit path
3      }
4  }
5
6  // look for [<body...>body text</body...>]
7
8      began = LCtext.indexOf("<body");
9      if (began > -1)
10     {
11         int body = LCtext.indexOf(">",began)+1;    // start of body
12         if (body > 1 && body < LCtext.length())
13         {
14             // parse body text attributes
15
16             String tagText = LCtext.substring(began+1,body-1);
17
18             ez_HTML.setLinkColor( _o_tg_db.parseColorValue(
19 _o_tg_db.getTagAttribString( tagText, "link" ), 0x0000ff));
20             ez_HTML.setForeground( _o_tg_db.parseColorValue(
21 _o_tg_db.getTagAttribString( tagText, "text" ), ez_HTML.getForeColor().getRGB()));
22             ez_HTML.setBackgroundColor( _o_tg_db.parseColorValue(
23 _o_tg_db.getTagAttribString( tagText, "bgcolor"), ez_HTML.getBackColor().getRGB()));
24
25             String fileName = _o_tg_db.getTagAttribString(tagText,"background");
26
27             if (fileName != null)
28             {
29                 ez_HTML.setBackImage(ez_HTML.getImageDB().fetchImage(fileName));
30             }
31
32             ez_HTML.setForeground(ez_HTML.getForeColor());
33             ez_HTML.setBackground(ez_HTML.getBackColor());
34
35             ended = LCtext.indexOf("</body",body);
36             if (ended > -1)
37             {
38                 return newText.substring(body,ended);    // nice formatting
39             }
40         }
41     }
42     return newText;
43 }
44
45 public static String lineCanon(String itsText; boolean isHTML)
46 {
47     if (itsText == null
48         || itsText.length() == 0) return itsText;
49
50     String newSeparator = isHTML? BL: PR;            // target return code
51     String searchSeparators[] =                     // replacable return
52 codes      {
53             "\n"+"\\r"                                // chr(10), chr(13)
54             , "\r"+"\\n"                                // chr(13), chr(10)
55
56             , "\\r"                                    // chr(13)
57             , "\\n"                                    // chr(10)
58             , "\\f"                                    // chr(12)
59             // , "\\t"                                    // chr(??)
60         };
61     String newText = new String(itsText);
62
63     for (int i = 0; i < searchSeparators.length; i++)    // replace old with new
64     {
65         String oldSeparator = new String(searchSeparators[i]);
66         newText = replaceString(newText,oldSeparator,newSeparator);
67     }
68
69     int oldLen = newText.length(), newLen = 0;

```

```

1      if (isHTML) do                                // prune double spaces &&
2      paragraph codes
3      {
4          oldLen = newText.length();
5          newText = replaceString(newText,BL+BL,BL);
6          newText = replaceString(newText,PR+PR,PR);
7          newLen = newText.length();
8      }
9      while (oldLen > newLen);
10     else
11         newText = replaceString(newText,"\\t","");
12
13     return newText;
14 }
15
16 private static String replaceString
17 (   String theString                                // original string
18   ,   String oldString                              // replace-ee
19   ,   String newString                              // replace-or
20 )
21 {
22     char[] tag = new char[1];
23     String piece = MT, theTag = MT;
24     StringBuffer out = new StringBuffer();
25
26     int oldIndex = -1, oldEnd = 0, n = 0;
27     int oldLength = oldString.length();
28
29     boolean newWhite = newString.equals(BL);
30
31     while ((oldIndex = theString.indexOf(oldString, oldEnd)) > -1)
32     {
33         piece = theString.substring(oldEnd,oldIndex); // got it? get it!
34         good... out.append(piece);                    // add the piece before
35                                                         the next old string
36
37         // don't convert returns into white space after tags (browser fashion)
38
39         if (newWhite && (n = out.length()) > 0)
40         {
41             tag[0] = out.charAt(n-1);
42             theTag = new String(tag);
43         }
44
45         if (!(newWhite && theTag.toLowerCase().equals(">")))
46             out.append(newString);                    // get the replacement
47
48         oldEnd = oldIndex + oldLength;                 // skip to the end of
49         this old string
50     }                                                  // nothing more to
51                                                         replace
52     if (oldEnd < theString.length())
53     {
54         piece = theString.substring(oldEnd);
55         out.append(piece);                            // add the piece after
56         the last old string
57     }
58     return out.toString();
59 }
60
61 /**
62  * %W% %E% Everett Stoub
63  *
64  * Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
65  *
66  * object with static methods to parse html tags
67  *
68  * this class is not downloaded unless needed

```

```

1  *
2  * @author Everett Stoub %I%, %G%
3  */
4  import java.awt.*;
5
6  public class _o_tg_db extends Object
7  {
8      public static final int getTagAlignment
9      (   String tagText
10         ,   int itsDefault
11         )
12      {
13          return getTagAlignment(tagText, _p_rs_db.AL, itsDefault);
14      }
15
16      public static final int getTagAlignment
17      (   String tagText
18         ,   String attrib
19         ,   int itsDefault
20         )
21      {
22          String itsAlignment = getTagAttribString(tagText, attrib);
23          if (itsAlignment != null)
24          {
25              itsAlignment = itsAlignment.toLowerCase();
26              if (itsAlignment.equals("left"))    return _p_rs_db.LEFT;
27              if (itsAlignment.equals("center"))  return _p_rs_db.CENTER;
28              if (itsAlignment.equals("right"))   return _p_rs_db.RIGHT;
29              if (itsAlignment.equals("top"))      return _p_rs_db.TOP;
30              if (itsAlignment.equals("middle"))  return _p_rs_db.MIDDLE;
31              if (itsAlignment.equals("bottom"))  return _p_rs_db.BOTTOM;
32          }
33          return itsDefault;
34      }
35
36      public static final int getTagAttribInt
37      (   String tagText
38         ,   String attrib
39         ,   int itsDefault
40         )
41      {
42          String value = getTagAttribString(tagText, attrib);
43          if (value == null)
44              return itsDefault;
45          else
46              return parseInt(value, itsDefault);
47      }
48
49      public static final String getTagAttribString
50      (   String itsText
51         ,   String attrib
52         )
53      {
54          String tagText = itsText.toLowerCase();
55          int i = tagText.indexOf(attrib), L = attrib.length();    // look for source
56          attribute
57          if (i > -1)
58          {
59              int a = tagText.indexOf(_p_rs_db.EQ, i + L) + 1;    // look for "="
60              assignment command
61              if (0 < a && a < i + L + 2)                          // avoid assignment
62              confusion
63              {
64                  String search = "\"' \\t\\n\\r>";                // quotes position <
65                  2
66                  int j = a, m = 0;                                // start after "="
67                  character
68                  int n = tagText.length();
69                  while (j < n                                       // skip single quote
70                  or whitespace or close tag

```

```

1      && (m = search.indexOf(tagText.charAt(j))) > 0) j++;
2      if (j < n)
3      {
4          if (m == 0)                                // double quote:
5      match for balance
6          {
7              m = tagText.indexOf(search.charAt(m),j+1);
8              if (m > -1) return itsText.substring(j+1,m);
9          }
10
11      // attribute is not enclosed in double quotes: use whitespace to
12      delimit
13
14          int k = j+1;
15          while (k < n                                // skip until any
16      quote or whitespace or close tag
17          && search.indexOf(tagText.charAt(k)) == -1) k++;
18
19          return itsText.substring(j,k).trim();
20      }
21  }
22  }
23  return null;
24  }
25
26  public static final Color getTagColor
27  (   String tagText // lower case
28      ,   int defaultRGB
29  )
30  {
31      return parseColorValue(getTagAttribString(tagText, "color"), defaultRGB);
32  }
33
34  public static final int getTagFontIndex
35  (   String tagText
36      ,   int defaultIndex
37  )
38  {
39      String sizeString = getTagAttribString(tagText, _p_rs_db.SZ);
40      if (sizeString == null)
41          return defaultIndex;
42
43      int itsSizeIndex = defaultIndex;
44
45      if (sizeString.startsWith("+"))
46          try {itsSizeIndex = defaultIndex +
47      Integer.parseInt(sizeString.substring(1));}
48          catch (NumberFormatException nf) {}
49      else
50          if (sizeString.startsWith("-"))
51              try {itsSizeIndex = defaultIndex -
52      Integer.parseInt(sizeString.substring(1));}
53              catch (NumberFormatException nf) {}
54          else
55              try {itsSizeIndex = Integer.parseInt(sizeString);}
56              catch (NumberFormatException nf) {}
57
58      return Math.max(Math.min(itsSizeIndex,16),0); // max index value for
59      readerFontSize array
60  }
61
62  public static final int getTagPerCentWidth(String tagText)
63  {
64      String width = getTagAttribString(tagText, "width");
65      if (width != null) try
66      {
67          if (width.endsWith("%"))
68              return -Integer.parseInt(width.substring(0,width.length()-1));
69          else
70              return Integer.parseInt(width);

```

```

1      }
2      catch (NumberFormatException nf)
3      {
4      }
5      return 0;
6  }
7
8  public static final int getTagWidth
9  (   String tagText
10     ,   int defaultWidth
11 )
12 {
13     int itsWidth = defaultWidth;
14     String width = getTagAttribString(tagText, "width");
15     if (width != null) try
16     {
17         if (width.endsWith("%"))
18             itsWidth = Integer.parseInt(width.substring(0,width.length()-
19 1))*defaultWidth/100;
20         else
21             itsWidth = Integer.parseInt(width);
22
23         itsWidth = Math.min(itsWidth,defaultWidth);
24     }
25     catch (NumberFormatException nf) {}
26     return itsWidth;
27 }
28
29 public static final Color parseColorValue
30 (   String colorValue
31     ,   int defaultRGB
32 )
33 {
34     int rgb = defaultRGB;
35     if (colorValue != null)
36     {
37         if (colorValue.startsWith("#"))
38         {
39             try {rgb = Integer.parseInt(colorValue.substring(1), 16);} catch
40 (NumberFormatException nf) {}
41         }
42         else    // try to interpret the named color
43         {
44             if (colorValue.equals("black"))    rgb = 0x000000; else
45             if (colorValue.equals("red"))      rgb = 0xff0000; else
46             if (colorValue.equals("green"))    rgb = 0x00ff00; else
47             if (colorValue.equals("blue"))     rgb = 0x0000ff; else
48             if (colorValue.equals("yellow"))   rgb = 0xffff00; else
49             if (colorValue.equals("magenta"))  rgb = 0xff00ff; else
50             if (colorValue.equals("cyan"))     rgb = 0x00ffff; else
51             if (colorValue.equals("white"))    rgb = 0xffffffff; else
52             if (colorValue.equals("lightGray")) rgb = 0xc0c0c0; else
53             if (colorValue.equals("gray"))     rgb = 0x808080; else
54             if (colorValue.equals("darkGray")) rgb = 0x404040; else
55             if (colorValue.equals("pink"))     rgb = 0xffafaf; else
56             if (colorValue.equals("orange"))   rgb = 0xffc800; else
57
58             // maybe a hex number after all (w/o preceeding # sign)
59
60             try {rgb = Integer.parseInt(colorValue, 16);} catch
61 (NumberFormatException nf) {}
62         }
63     }
64     return new Color(rgb);
65 }
66
67 public static final int parseInt
68 (   String value
69     ,   int itsDefault
70 )

```

```

1      {
2          try
3          {
4              return Integer.parseInt(value);
5          }
6          catch (NumberFormatException nf)
7          {
8              return itsDefault;
9          }
10     }
11 }
12
13 /**
14  *  %W% %E% Everett Stoub
15  *
16  *  Copyright (c) '96-'97 ICON Web Products, Inc. All Rights Reserved.
17  *
18  *  a pagable panel to display multicolumn text and graphics
19  *
20  *  HTML tag parsing honors the following commands and attributes
21  *
22  *      • <a>...</a>                anchor tag
23  *          href                    relative or absolute url for hyperlink
24  *
25  *      • <address>...</address>    address (new italic paragraph)
26  *
27  *      • <applet>...</applet>      applet tag & range: range is hidden, applet is
28  *      not launched
29  *
30  *      • <b>...</b>                bold
31  *
32  *      • <basefont>...</basefont>  basic font size for document; </basefont>
33  *      reverts to 4
34  *          size=value              relative or absolute index size adjustment
35  *
36  *      • <big>...</big>            increase font size one step
37  *
38  *      • <blockquote>...</blockquote>  blockquote (left & right margin indentation)
39  *
40  *      • <br>...                    new paragraph, no extra halfspace
41  *          align=type              left, center, right
42  *
43  *      • <center>...</center>      center justify
44  *
45  *      • <cite>...</cite>          citation (italic)
46  *
47  *      • <code>...</code>          code (monospaced)
48  *
49  *      • <dir>...</dir>            unordered directory list
50  *          type=bullet type        circle, square, or disc
51  *
52  *      • <dd>...</dd>              definition list definition component
53  *
54  *      • <dl>...</dl>              unordered (non-bulleted) definition list
55  *
56  *      • <dt>...</dt>              definition list term component
57  *
58  *      • <em>...</em>              emphasis (italic)
59  *
60  *      • <font>...</font>          font color and size; </font> reverts to normal
61  *          color=color             rgb or color name
62  *          size=value              size index value
63  *
64  *      • <form>...</form>          form specification domain
65  *          action=url              server contact for response
66  *          enctype=encoding        element value coding system
67  *          method=style            either get or post
68  *
69  *      • <hr>                      horizontal rule
70  *          align=type              left, center, right

```

1	*	size=pixels	vertical rule size
2	*	noshade	sets flat rounded rect rule
3	*	width=value or %	specify width up to column width value
4	*		
5	*	• <h1...6>...</h1...6>,	heading levels 1 thru 6
6	*	align=type	left, center, right
7	*		
8	*	• <i>...</i>	italic
9	*		
10	*	• 	insert .gif or .jpeg image
11	*	align=type	bottom, middle, or top
12	*	src=url	image url
13	*		
14	*	• <input>	form input element specification
15	*	align=type	top, middle, or bottom (image)
16	*	border=n	pixel border thickness (image)
17	*	checked	marks radio or checkbox initially selected
18	*	maxlength=n	max # of chars to accept (file or
19		password)	
20	*	name=name	passed to designated url
21	*	size=n	max # of chars to display (file or
22		password)	
23	*	src=url	image url
24	*	type=type	button, checkbox, file, hidden, image,
25		password, radio, reset, submit, text	
26	*	value=string	passed to designated url
27	*		
28	*	• <kbd>...</kbd>	keyboard (monospaced)
29	*		
30	*	• ...	list item
31	*	type=bullet type	for unordered: circle, square, or disc
32	*	type=number type	for ordered: A, a, I, i, or 1
33	*	start=int	overriding ordered list number
34	*		
35	*	• <menu>...</menu>	unordered menu list
36	*	type=bullet type	circle, square, or disc
37	*		
38	*	• ...	ordered (nested) list
39	*	type=number type	A, a, I, i, or 1
40	*	start=int	overriding ordered list number
41	*		
42	*	• <option>...</option>	pop-up input list element definition
43	*	selected	initially select element
44	*	value=string	passed to designated url (instead of
45		<option> contents)	
46	*		
47	*	• <p>...</p>	new paragraph, extra halfspace
48	*	align=type	left, center, right
49	*		
50	*	• <plaintext>	plaintext (rendered literally, with monospaced
51		font): no terminal	
52	*		
53	*	• <pre>...</pre>	pre-formatted text (rendered literally, with
54		monospaced font)	
55	*		
56	*	• <select>...</select>	pop-up input list specification domain
57	*	multiple	allow multiple selections
58	*	name=name	passed to designated url
59	*	size=n	number of items to display
60	*		
61	*	• <small>...</small>	decrease font size one step
62	*		
63	*	• <strike>...</strike>	strikethrough text
64	*		
65	*	• ...	strong (bold)
66	*		
67	*	• _{...}	subscript
68	*		
69	*	• ^{...}	superscript
70	*		


```

1  *   • <table>...</table>           table tag & range (replaced with hot image)
2  *
3  *   • <textarea>...</textarea>      text input area specification domain
4  *       cols=n                      number of characters to display in a row
5  *       name=name                   passed to designated url
6  *       rows=n                     number of lines to display in area
7  *
8  *   • <tt>...</tt>                 teletype (monospaced)
9  *
10 *   • <u>...</u>                   underline
11 *
12 *   • <ul>...</ul>                 unordered (nested) list
13 *       type=bullet type           circle, square, or disc
14 *
15 *   • <var>...</var>               variable (italic)
16 *
17 *   • <!--...-->                   html comments (stripped out)
18 *
19 * @author Everett Stoub %I%, %G%
20 */
21 import java.awt.*;
22 import java.awt.image.*;
23 import java.applet.*;
24 import java.net.*;
25 import java.io.*;
26 import java.util.*;
27
28 public final class _p_rs_db extends Panel
29 {
30     public final static int    LEFT        = 0;    // position
31     public final static int    CENTER      = 1;    // position
32     public final static int    RIGHT       = 2;    // position
33     public final static int    TOP         = 3;    // position
34     public final static int    MIDDLE      = 4;    // position
35     public final static int    BOTTOM     = 5;    // position
36     public final static int    POSITION     = 7;    // position mask
37     public final static int    UNDERLINE  = 8;    // text
38     public final static int    STRIKETHRU  = 16;   // text
39     public final static int    SUBSCRIPT   = 32;   // text
40     public final static int    SUPSCRIPT   = 64;   // text
41     public final static int    HYPERLINK   = 128;  // text
42     public final static int    LEFTINDENT  = 256;  // text
43     public final static int    RIGHTINDENT = 512;  // text
44     public final static int    HIDDENTEXT  = 1024; // text
45     public final static int    UNORDERLIST = 2048; // text
46     public final static int    ORDEREDLIST = 4096; // text
47
48     public final static String AD          = "address";
49     public final static String AL          = "align";
50     public final static String AN          = "a";
51     public final static String AP          = "applet";
52     public final static String BD          = "b";
53     public final static String BF          = "basefont";
54     public final static String BG          = "big";
55     public final static String BL          = " ";
56     public final static String BQ          = "blockquote";
57     public final static String BR          = "br";
58     public final static String CD          = "code";
59     public final static String CJ          = "center";
60     public final static String CK          = "checked";
61     public final static String CR          = "Courier";
62     public final static String CT          = "cite";
63     public final static String ID          = "dd";
64     public final static String IL          = "dl";
65     public final static String DT          = "dt";
66     public final static String IR          = "dir";
67     public final static String EM          = "em";
68     public final static String EN          = "elementNumber"; // private form
69     element attribute
70     public final static String EQ          = "=";

```

```

1      public final static String FM      = "form";           // private form tage
2      public final static String FN      = "formNumber";      // private form
3      attribute
4      public final static String FS      = "TimesRoman";
5      public final static String FT      = "font";
6      public final static String HF      = "href";
7      public final static String HN      = "h";
8      public final static String HR      = "hr";
9      public final static String IM      = "img";
10     public final static String IN      = "input";
11     public final static String IT      = "i";
12     public final static String KB      = "kbd";
13     public final static String LI      = "li";
14     public final static String ML      = "menu";
15     public final static String MT      = "";
16     public final static String MX      = "maxlength";
17     public final static String NM      = "name";
18     public final static String OL      = "ol";
19     public final static String OP      = "option";
20     public final static String PA      = "p";
21     public final static String PF      = "<";
22     public final static String PR      = "pre";
23     public final static String PT      = "plaintext";
24     public final static String SB      = "sub";
25     public final static String SF      = ">";
26     public final static String SG      = "strong";
27     public final static String SK      = "strike";
28     public final static String SL      = "/";
29     public final static String SM      = "small";
30     public final static String SP      = "sup";
31     public final static String SR      = "src";
32     public final static String ST      = "select";
33     public final static String SZ      = "size";
34     public final static String TA      = "textarea";
35     public final static String TB      = "tblt";           // private table tag
36     public final static String TP      = "type";
37     public final static String TT      = "tt";
38     public final static String UL      = "ul";
39     public final static String UN      = "u";
40     public final static String VA      = "var";
41     public final static String VL      = "value";
42     public final static String ZI      = "Release for fullsize image view...";
43     public final static String ZM      = "Click to zoom image...";
44     public final static String ZN      = "Release for new reader screen...";
45     public final static String ZW      = "Wait for new reader screen...";
46     public final static String ZX      = "Wait for fullsize image view...";
47     public final static String ZZ      = "... ..";
48
49     public ez_html      ez_HTML      = null;
50     private _f_rs_db    frame      = null;
51     private _o_nt_db    charEntConverter = null;
52
53     public String      readerContent      = null;
54     public Insets      insets            = new Insets(0,0,0,0);
55     private int        maxColumns        = 15;
56     private Dimension  minimumSize       = new Dimension( 0, 0);
57     private Dimension  preferredSize     = new Dimension(-1,-1);
58     private Image      readerScreen      = null;
59     public boolean     zoomable          = false;
60
61     public int          readerFontSize[]  = { 8,
62 9,10,11,12,14,18,24,30,36,42,48,56,64,72,80,96};
63     public int          readerFootIndex   = 2;           // this 2-step decrement is
64     also hard coded in stepFontSize()
65     public int          readerSizeIndex   = 4;           // default for on-webPage
66     article rendering
67     public int          readerEnWidth     = 6;
68     public Font         readerScreenFont  = null;
69     public Font         readerFooterFont  = null;
70     public int          footerTextHeight  = 12;

```

```

1
2     public int     readerNumCols     = 1;
3     public int     readerLeading      = 0;
4     public int     readerPageNum     = 1;
5     public int     readerNumPages    = 0;
6     public int     readerEstPages    = 1;
7     public int     readerEndIndex     = 1;
8     public int     readerStartIndex   = 0;
9     public int     readerTextLength   = 0;
10    public boolean  readerScreenDirty = true;           // true if panel
11    needs redraw
12    public boolean  readerFrame       = false;         // true if panel
13    owned by frame
14    public boolean  readerHeader      = false;         // true if article
15    begins with `title
16    public boolean  readerMarker      = false;         // true to present
17    markers
18    public boolean  readerFooter      = false;         // true to present
19    footer
20
21    public Vector   readerPageImage   = new Vector(50);
22    public Vector   readerListAccts   = new Vector(50);
23    public Vector   readerLinkHeads   = new Vector(50);
24    public Vector   readerLinkRects   = new Vector(50);
25    public Vector   readerZoomHeads   = new Vector(50);
26    public Vector   readerZoomRects   = new Vector(50);
27    public Vector   readerPageComps   = new Vector(50);
28    public Vector   readerPageStart   = new Vector(50);
29    public Vector   readerFontStart   = new Vector(50);
30    public Vector   readerLinkStart   = new Vector(50);
31    public Vector   readerTintStart   = new Vector(50);
32    public Vector   readerTypeStart   = new Vector(50);
33
34    public Rectangle readerRect        = new Rectangle();
35
36    // variables principally for paintArea
37
38    public int     border      = 0;
39    public int     padding     = 0;
40    public int     spacing     = 0;
41
42    public int     maxWidth    = 0;
43    public int     maxHeight   = 0;
44    public int     controlSize = 8;
45
46    public boolean  firstWord   = true;           // current setting
47    public boolean  lastColumn  = false;          // current setting
48    public boolean  lastRow     = false;          // current setting
49    public boolean  termDefinition = false;
50    public boolean  plaintext    = false;         // true blocks tag
51    parsing
52
53    public Rectangle aheadRect      = new Rectangle();
54
55    public boolean  newListItem     = false;       // current page
56    pending item
57    public Vector   listAccount     = new Vector(); // current page
58    (nested) list elements
59    public Vector   linkHeads       = new Vector(); // current page
60    public Vector   linkRects      = new Vector(); // current page
61    public Vector   zoomHeads      = new Vector(); // current page
62    public Vector   zoomRects      = new Vector(); // current page
63    public Vector   pageComps      = new Vector(); // current page
64
65    public Font     textFont        = null;        // current setting
66    public FontMetrics textMetrics   = null;        // current setting
67    public Color     textColor      = null;        // current setting
68    public int       textHeight     = 0;           // current setting
69    public int       textHalfSpace  = 0;           // current setting
70    public int       textAlign      = LEFT;        // current setting

```

```

1
2     public int         colNum = 0, colWidth = 0, colSpacing = 0, numColumns = 0,
3     tagLength = 0;
4     public int         endIndex = 0, nextWidth = 0, xb = 0, te = 0, xe = 0, yb = 0,
5     ye = 0;
6
7     public boolean     markings          = false;           // current word
8
9     public boolean     underlining       = false;           // current word
10    public boolean     strikethrough     = false;           // current word
11    public boolean     subscript         = false;           // current word
12    public boolean     superscript       = false;           // current word
13    public boolean     hyperlink         = false;           // current word
14    public boolean     leftIndent        = false;           // current word
15    public boolean     rightIndent       = false;           // current word
16    public boolean     hidingText        = false;           // current word
17    public boolean     unorderedList     = false;           // current word
18    public boolean     orderedList       = false;           // current word
19    public URL          hyperHead        = null;            // current word
20
21    public Vector       lineImages        = new Vector();    // current row
22    public Vector       dataValues        = new Vector();    // current row
23    public Vector       wordBlocks        = new Vector();    // current row
24    public Vector       hyperLinks       = new Vector();    // current row
25    public Vector       styleCodes       = new Vector();    // current row
26    public Vector       wordColors       = new Vector();    // current row
27    public Vector       charsFonts       = new Vector();    // current row
28
29    public final boolean addBlock(String tagBlock, Graphics g, boolean count)
30    {
31        //
32        dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
33        aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").addBlock("+tagBloc
34        k+"");
35
36        if (tagBlock == null)
37            return true;
38
39        StringTokenizer words =
40            new StringTokenizer(tagBlock,BL,true);           // "true" returns blanks (for
41        counting)
42        String nextWord = MT;
43
44        while (words.hasMoreTokens())
45        {
46            nextWord = words.nextToken();
47            nextWidth = addWord(nextWord, false, count);
48
49            if (te + nextWidth < xe - indent(true))          // it will fit on this line,
50        so add it on
51            {
52                te += nextWidth;
53            }
54            else                                              // it won't fit if added, so
55        draw the line
56            {
57                endIndex += paintRow(g, 1);                  // try to punch out prior
58        pending line elements
59                te = xb + indent(false);
60                if (yb > ye)                                  // start a new column?
61                {
62                    if (lastColumn) return false;           // no more columns left to
63        fill
64                    nextColumn();
65                }
66                if (nextWord.equals(BL))                     // drop leading blank at new
67        line
68                {
69                    endIndex++;
70                    clearRowElements();

```

```

1      savePageData
2      (   readerPageNum
3          ,   endIndex
4      );
5
6      nextWord = MT;
7      nextWidth = MT.length();
8
9      te += nextWidth;
10     possibly, a monster!
11
12     if (te > xe - indent(true) && preferredSize.width > 0)
13     {
14         preferredSize.width = Math.max
15         (   insets.left + insets.right + 2
16             +   nextWidth
17             ,   preferredSize.width
18         );
19
20         //
21         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
22 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").addBlock(); new
23 preferredSize("+preferredSize.width+", "+preferredSize.height+") = resize() +
24 ("+(size().width - preferredSize.width)+", "+(size().height - preferredSize.height)+"):
25 nextWord = ["+nextWord+"]");
26     }
27     while (te > xe - indent(true))
28     breaking the word
29     {
30         te = xb + indent(false);
31         String monster = nextWord;
32         place for a hyphenator
33         nextWord = MT;
34         int e = monster.length();
35         while (te + nextWidth > xe - indent(true) && e > 0)
36         {
37             until fit (one line)
38                 clearRowElements();
39                 nextWord = monster.substring(e-1,e) + nextWord;
40                 monster = monster.substring(0,e-1);
41                 // dbg("... monster = ["+monster+"], nextWord = ["+nextWord+"]");
42                 e = monster.length();
43                 nextWidth = addWord(monster, true, count);
44         }
45         te += nextWidth;
46         endIndex += paintRow(g, 0);
47         line elements
48         te = xb + indent(false);
49         if (yb > ye)
50         {
51             if (lastColumn) return false;
52             fill
53                 nextColumn();
54         }
55         nextWidth = addWord(nextWord, false, count);
56         te += nextWidth;
57     }
58 }
59 }
60 return true;
61 block
62 }
63
64 private final int addTag(String itsText, Graphics g)
65 {
66     //
67     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
68 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").addTag("+itsText+"
69 )");
70

```

```

1      String tagText = itsText.toLowerCase();
2      int addIndex = 0;
3      tagLength = tagText.length() + 2;
4
5      if (tagText.equals(PA)                                // an html new paragraph
6  tag w/o attributes
7      || tagText.startsWith(PA+BL)                          // an html new paragraph
8  tag w/ attributes
9      || tagText.startsWith(BR)                             // an html new line tag
10     || tagText.equals(AD))                                 // an html address tag
11     {
12         addIndex += paintRow(g,0);                          // try to punch out
13     pending line elements
14         if (!startNewParagraph(tagText.startsWith(BR)?0:1)) return addIndex;
15         textAlign =
16         _o_tg_db.getTagAlignment(tagText, (tagText.startsWith(PA)?LEFT:textAlign));
17         if (tagText.equals(AD)) this.setFont(new
18         Font(textFont.getName(),textFont.getStyle() + Font.ITALIC,textFont.getSize()));
19     }
20     else if (tagText.equals(SL+PA)                          // an html end paragraph
21 tag
22     || tagText.equals(SL+AD))                              // an html end address
23 tag
24     {
25         addIndex += paintRow(g,0);                          // try to punch out
26     pending line elements
27         textAlign = LEFT;                                    // default alignment
28         if (tagText.equals(SL+AD)) this.setFont(new
29         Font(textFont.getName(),textFont.getStyle() - Font.ITALIC,textFont.getSize()));
30     }
31     else if (tagText.startsWith(IM))                        // an html image tag
32     {
33         int newAddIndex = _o_im_db.placeImage(tagText, ez_HTML, this, ++addIndex,
34 g); // bias for sign detect
35         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
36         // reverse bias
37     }
38     else if (tagText.startsWith(HR))                        // an html horizontal
39 rule tag
40     {
41         int newAddIndex = _o_hr_db.placeRule(tagText, this, ++addIndex, g);
42         // bias for sign detect
43         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
44         // reverse bias
45     }
46     else if (tagText.startsWith(TB))                        // a preprocessed html
47 table tag
48     {
49         int newAddIndex = _o_tb_db.placeTable(tagText, ez_HTML, this, ++addIndex,
50 g); // bias for sign detect
51         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
52         // reverse bias
53     }
54     else if (tagText.startsWith(HN))                        // an html heading level
55 tag
56         && (tagText.length() < 3                            // either without
57 attributes
58         || tagText.substring(2,3).equals(BL))              // or with attributes to
59 follow
60     {
61         addIndex += paintRow(g,0);                          // try to punch out
62     pending line elements
63         if (!startNewParagraph(1)) return addIndex;
64         textAlign = _o_tg_db.getTagAlignment(tagText, LEFT);
65         this.setFont(new
66         Font(textFont.getName(),Font.BOLD,readerFontSize[readerSizeIndex + 4 -
67         _o_tg_db.parseInt(tagText.substring(1,2), 4)]));
68     }
69     else if (tagText.startsWith(AN+BL))                     // an html anchor tag
70     {

```

```

1      try
2      {
3          hyperHead = new URL(ez_HTML.getDocumentBase(),
4      _o_tg_db.getTagAttribString(tagText,HF));
5          hyperLink = true;
6          textColor = ez_HTML.linkColor;
7      }
8      catch (MalformedURLException m) {};
9      }
10     else if (tagText.equals(SL+AN))                // an html end anchor tag
11     {
12         if (hyperLink)
13         {
14             hyperHead = null;
15             hyperLink = false;
16             textColor = getForeground();
17         }
18     }
19     else if (tagText.equals(BD))                    // an html bold tag
20         || tagText.equals(SG))                    // an html strong tag
21     {
22         this.setFont(new Font(textFont.getName(),textFont.getStyle() +
23 Font.BOLD,textFont.getSize()));
24     }
25     else if (tagText.equals(SL+BD))                // an html end bold tag
26         || tagText.equals(SL+SG))                // an html end strong tag
27     {
28         this.setFont(new Font(textFont.getName(),textFont.getStyle() -
29 Font.BOLD,textFont.getSize()));
30     }
31     else if (tagText.equals(IT))                    // an html italic tag
32         || tagText.equals(CT))                    // an html citation tag
33         || tagText.equals(EM))                    // an html emphasis tag
34         || tagText.equals(VA))                    // an html variable tag
35     {
36         this.setFont(new Font(textFont.getName(),textFont.getStyle() +
37 Font.ITALIC,textFont.getSize()));
38     }
39     else if (tagText.equals(SL+IT))                // an html end italic tag
40         || tagText.equals(SL+CT))                // an html end citation
41 tag        || tagText.equals(SL+EM))                // an html end emphasis
42 tag        || tagText.equals(SL+VA))                // an html end variable
43 tag
44 tag
45 tag
46     {
47         this.setFont(new Font(textFont.getName(),textFont.getStyle() -
48 Font.ITALIC,textFont.getSize()));
49     }
50     else if (tagText.startsWith(FT))                // an html font tag
51     {
52         this.setFont(new
53 Font(textFont.getName(),textFont.getStyle(),readerFontSize[_o_tg_db.getTagFontIndex(ta
54 gText,readerSizeIndex)],_o_tg_db.getTagColor(tagText,textColor.getRGB()));
55     }
56     else if (tagText.equals(SL+FT))                // an html end font tag
57         || (tagText.startsWith(SL+HN))            // an html end heading
58 tag        && tagText.length() < 4))
59     {
60         this.setFont(readerScreenFont,getForeground());
61         if (tagText.startsWith(SL+HN)                // an html end heading
62 level tag        && tagText.length() < 4)
63         {
64             addIndex += paintRow(g,0);                // try to punch out
65             pending line elements
66             if (!startNewParagraph(0)) return addIndex;
67         }
68     }
69 }
70 }

```

```

1         else if (tagText.startsWith(LI))                // an html list item tag
2             || tagText.equals(DT)                        // an html definition
3     term tag
4             || tagText.equals(DD))                        // an html definition
5     component tag
6     {
7         int newAddIndex = _o_li_db.addItem(tagText, itsText, this, ++addIndex, g);
8         // bias for sign detect
9         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
10        // reverse bias
11    }
12    else if (tagText.startsWith(UL))                        // an html unordered list
13    tag
14        || tagText.startsWith(OL)                        // an html ordered list
15    tag
16        || tagText.startsWith(ML)                        // an html menu list tag
17        || tagText.equals(DL)                            // an html definition
18    list tag
19        || tagText.startsWith(DR))                        // an html directory list
20    tag
21    {
22        int newAddIndex = _o_li_db.addList(tagText, itsText, this, ++addIndex, g);
23        // bias for sign detect
24        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
25        // reverse bias
26    }
27    else if (tagText.equals(SL+UL))                        // an html end unordered
28    list tag
29        || tagText.equals(SL+OL)                        // an html end ordered
30    list tag
31        || tagText.equals(SL+ML)                        // an html end menu list
32    tag
33        || tagText.equals(SL+DL)                        // an html end definition
34    list tag
35        || tagText.equals(SL+DR))                        // an html end directory
36    list tag
37    {
38        int newAddIndex = _o_li_db.endList(tagText, itsText, this, ++addIndex, g);
39        // bias for sign detect
40        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
41        // reverse bias
42    }
43    else if (tagText.startsWith(IN))
44    {
45        int newAddIndex = _o_fm_db.addInput(tagText, itsText, this, ++addIndex,
46    g); // bias for sign detect
47        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
48        // reverse bias
49    }
50    else if (tagText.startsWith(OP))
51    {
52        int newAddIndex = _o_fm_db.addOption(tagText, itsText, this, ++addIndex,
53    g); // bias for sign detect
54        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
55        // reverse bias
56    }
57    else if (tagText.equals(SL+OP))
58    {
59        int newAddIndex = _o_fm_db.endOption(tagText, itsText, this, ++addIndex,
60    g); // bias for sign detect
61        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
62        // reverse bias
63    }
64    else if (tagText.startsWith(ST))
65    {
66        int newAddIndex = _o_fm_db.addSelector(tagText, itsText, this, ++addIndex,
67    g); // bias for sign detect
68        if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
69        // reverse bias
70    }

```



```

1      else if (tagText.equals(SL+ST))
2      {
3          int newAddIndex = _o_fm_db.endSelector(tagText, itsText, this, ++addIndex,
4 g); // bias for sign detect
5          if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
6          // reverse bias
7      }
8      else if (tagText.startsWith(TA))
9      {
10         int newAddIndex = _o_fm_db.addTextArea(tagText, itsText, this, ++addIndex,
11 g); // bias for sign detect
12         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
13         // reverse bias
14     }
15     else if (tagText.equals(SL+TA))
16     {
17         int newAddIndex = _o_fm_db.endTextArea(tagText, itsText, this, ++addIndex,
18 g); // bias for sign detect
19         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
20         // reverse bias
21     }
22     else if (tagText.startsWith(FM))
23     {
24         int newAddIndex = _o_fm_db.addForm(tagText, itsText, this, ++addIndex, g);
25         // bias for sign detect
26         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
27         // reverse bias
28     }
29     else if (tagText.equals(SL+FM))
30     {
31         int newAddIndex = _o_fm_db.endForm(tagText, itsText, this, ++addIndex, g);
32         // bias for sign detect
33         if (newAddIndex < 0) return ++newAddIndex; else addIndex = --newAddIndex;
34         // reverse bias
35     }
36     else if (tagText.equals(BG) // an html big tag
37             || tagText.equals(SL+SM) // an html end small tag
38             || tagText.equals(SM) // an html small tag
39             || tagText.equals(SL+BG)) // an html end big tag
40     {
41         this.setFont(new
42 Font(textFont.getName(),textFont.getStyle(),readerFontSize[Math.max(Math.min(getFontIn
43 dex(textFont,readerSizeIndex)+((tagText.equals(BG) || tagText.equals(SL+SM))?1):(-
44 1)),readerFontSize.length - 1),0));
45     }
46     else if (tagText.equals(BQ) // an html blockquote tag
47             || tagText.equals(SL+BQ)) // an html end blockquote
48 tag
49     {
50         leftIndent = rightIndent = tagText.equals(BQ);
51     }
52     else if (tagText.equals(CJ) // an html center tag
53             || tagText.equals(SL+CJ)) // an html end center tag
54     {
55         textAlign = tagText.equals(CJ)?CENTER:LEFT;
56     }
57     else if (tagText.equals(SK) // an html strikethrough
58 tag
59             || tagText.equals(SL+SK)) // an html end
60 strikethrough tag
61     {
62         strikethrough = tagText.equals(SK);
63     }
64     else if (tagText.equals(UN) // an html underline tag
65             || tagText.equals(SL+UN)) // an html end underline
66 tag
67     {
68         underlining = tagText.equals(UN);
69     }
70     else if (tagText.equals(SB) // an html subscript tag

```

```

1      || tagText.equals(SL+SB)) // an html end subscript
2  tag
3      {
4          subscript = tagText.equals(SB);
5      }
6      else if (tagText.equals(SP)) // an html superscript
7  tag
8      || tagText.equals(SL+SP)) // an html end
9  superscript tag
10     {
11         subscript = tagText.equals(SP);
12     }
13     else if (tagText.equals(CD)) // an html code tag:
14  monospaced font
15         || tagText.equals(PR) // an html pre tag:
16  monospaced font
17         || tagText.equals(PT) // an html plaintext tag:
18  monospaced font
19         || tagText.equals(TT) // an html teletype tag:
20  monospaced font
21         || tagText.equals(KB)) // an html keyboard tag:
22  monospaced font
23     {
24         this.setFont(new Font(CR,Font.PLAIN,textFont.getSize()));
25         if (tagText.equals(PT)) plaintext = true; // if true, further tag
26 parsing is blocked
27     }
28     else if (tagText.equals(SL+CD)) // an html end code tag:
29  standard font
30         || tagText.equals(SL+PR) // an html end pre tag:
31  standard font
32         || tagText.equals(SL+TT) // an html end teletype
33 tag: standard font
34         || tagText.equals(SL+KB)) // an html end keyboard
35 tag: standard font
36     {
37         this.setFont(new Font(FS,Font.PLAIN,textFont.getSize()));
38     }
39     else if (tagText.startsWith(BF)) // an html basefont tag
40         || tagText.equals(SL+BF)) // an html end basefont
41 tag
42     {
43         readerSizeIndex =
44 (tagText.startsWith(BF))?_o_tg_db.getTagFontIndex(tagText,readerSizeIndex):4;
45         readerScreenFont = new Font(FS, Font.PLAIN,
46 readerFontSize[readerSizeIndex]);
47         this.setFont(readerScreenFont);
48     }
49     else // whatever has not been
50 handled is ignored but counted
51     {
52         dbg("tag = <"+tagText+"> was not handled; tagLength = "+tagLength);
53     }
54
55     lineImages.addElement(null);
56     dataValues.addElement
57     ( new Rectangle
58         ( tagLength // "x" increment
59 for endIndex (character count)
60         , 0 // "y" for
61 vertical placement of image (ascent)
62         , 0 // "width" element
63 width on line
64         , 0 // "height" element
65 drop below baseline (descent)
66     )
67 );
68     wordBlocks.addElement(MT);
69     hyperLinks.addElement(null);
70

```

```

1      styleCodes.addElement(styleCode());
2      wordColors.addElement(textColor);
3      charsFonts.addElement(textFont);
4
5      return addIndex;                                // the endIndex increment
6  precipitated by this tag
7  }
8
9      private final int addWord(String nextWord, boolean addHyphen, boolean count)
10     {
11         //
12         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
13 aderContent.length()<30?ZZ.substring(readerContent.length()):MT')+").addWord("\ "+nextWo
14 rd+"\ ");
15
16         if (nextWord == null) return 0;                // nothing to draw
17
18         String block = charEntConverter.convertCharacterEntities(nextWord);
19         if (hidingText) block = MT;                    // cheap trip for hidden
20 text
21         if (addHyphen) block = block + "-";            // don't count this: it's
22 an added character
23
24         int wIm = textMetrics.stringWidth(block);      // all words are followed
25 by spaces
26         int hIm = textHeight + readerLeading;
27
28         if (firstWord && wIm > 0)
29         {
30             minimumSize = new Dimension
31             (    wIm + insets.left + insets.right + 2
32               ,    hIm + insets.top + insets.bottom + 2
33             );
34             //
35             dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
36 aderContent.length()<30?ZZ.substring(readerContent.length()):MT')+").addWord(): new
37 minimumSize("+minimumSize.width+", "+minimumSize.height+") for [" +nextWord+"]");
38
39             firstWord = false;
40         }
41
42         int maxAscent = (textHeight+2)*2/3 + readerLeading; // ascent and descent are
43 wrong in some browsers
44
45         lineImages.addElement(null);
46         dataValues.addElement
47         (    new Rectangle
48           (    (count?nextWord.length():0)                // "x"          increment
49 for endIndex (character count)
50             ,    maxAscent                                // "y"          for
51 vertical placement of image (ascent)
52             ,    wIm                                        // "width"      element
53 width on line
54             ,    hIm - maxAscent                          // "height"    element
55 drop below baseline (descent)
56         )
57     );
58     wordBlocks.addElement(new String(block));
59     hyperLinks.addElement(hyperHead);
60
61     styleCodes.addElement(styleCode());
62     wordColors.addElement(textColor);
63     charsFonts.addElement(textFont);
64
65     return wIm;                                         // the width of this
66 element on the line
67 }
68
69     public final int calculateReaderColumns()
70     {

```

```
1         return calculateReaderColumns(this.size().width);    // the width of this
2     panel
3     }
4
5     public final int calculateReaderColumns(int breadth)
6     {
7         //
8         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
9 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").calculateReaderCol
10 umns(): breadth = "+breadth);
11
12         int numCols = breadth / standardLineWidth();
13         // dbg("... breadth = "+breadth+" / numChar("+numChar+") /
14 readerEnWidth("+readerEnWidth+") = numCols("+numCols+"); maxColumns = "+maxColumns);
15
16         if (numCols < 1)
17             numCols = 1;
18
19         return Math.min(numCols, maxColumns);
20     }
21
22     public final void clearPage()
23     {
24         //
25         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
26 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").clearPage()");
27
28         newListItem = false;
29         listAccount = new Vector();
30
31         unorderedList = false;
32         orderedList = false;
33
34         hyperHead = null;
35         hyperLink = false;
36         linkHeads = new Vector();
37         linkRects = new Vector();
38
39         zoomHeads = new Vector();
40         zoomRects = new Vector();
41
42         removeAll();
43     }
44
45     public void clearRect(int x, int y, int width, int height)
46     {
47         // override, to retain transparency for background tiling if any
48     }
49
50     public final void clearRowElements()
51     {
52         lineImages.removeAllElements();
53         dataValues.removeAllElements();
54         wordBlocks.removeAllElements();
55         hyperLinks.removeAllElements();
56         styleCodes.removeAllElements();
57         wordColors.removeAllElements();
58         charsFonts.removeAllElements();
59     }
60
61     public Dimension columnSize()
62     {
63         return new Dimension(effectiveWidth(), readerRect.height);
64     }
65
66     private final void dbg(String s)
67     {
68         if (debug) System.out.println(s);
69     }
70     private boolean debug = true;    // Print debugging info?
```

```

1
2     public final void dispose()
3     {
4         //
5         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
6 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").dispose()");
7
8         if (readerScreen != null)
9             readerScreen.flush();
10    }
11
12    public final void doViewerSize(boolean smaller)
13    {
14        //
15        dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
16 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").doViewerSize("+sma
17 ller+") # comps = "+countComponents());
18
19        stepFontSize(smaller);
20
21        setReaderRect();
22        clearRowElements();
23        resetReaderView();
24        estimateNumPages();
25        repaintReaderScreen();
26    }
27
28    public int effectiveWidth()
29    {
30        if (!markings
31            || aheadRect.x == 0)
32            return colWidth - indent(true) - indent(false);
33        else
34            return Math.min
35                ( aheadRect.x + indent(true)
36                  ,   xb + colWidth
37                  ) - xb - indent(true) - indent(false);
38    }
39
40    public void estimateNumPages()
41    {
42        //
43        dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
44 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").estimateNumPages()
45 ");
46
47        if (readerNumPages == 0)
48        {
49            if (readerEndIndex == readerTextLength)
50            {
51                readerEstPages = readerNumPages = readerPageNum;
52                // dbg("... exact value = "+readerPageNum);
53            }
54            else
55                // estimate number of pages
56            {
57                readerEstPages = Math.round(0.5f
58                    + (float)readerPageNum
59                    * (float)readerTextLength
60                    / (float)readerEndIndex);
61                // dbg("... est'd value = "+readerEstPages);
62            }
63        }
64        else
65        {
66            readerEstPages = readerNumPages; // exact value is known
67            // dbg("... exact value = "+readerNumPages);
68        }
69    }
70    public final int getFontIndex(Font itsFont, int itsDefault)

```

```

1      {
2          int itsIndex = 0, itsSize = itsFont.getSize();
3
4          while (itsIndex < readerFontSize.length
5              && itsSize != readerFontSize[itsIndex]) itsIndex++;
6
7          if (itsIndex == readerFontSize.length)
8              itsIndex = readerSizeIndex;          // default
9
10         return itsIndex;
11     }
12
13     public final boolean imageUpdate(Image img, int flags, int x, int y, int w, int h)
14     {
15         //
16         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
17 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").imageUpdate(flags
18 = "+flags+"");
19
20         if
21         ( ( flags
22           & ( ImageObserver.ALLBITS
23             | ImageObserver.FRAMEBITS
24           )
25           ) != 0)    // this image is complete (at last)
26         {
27             Image pageImage = (Image)readerPageImage.elementAt(readerPageNum - 1);
28             if (pageImage != null)
29                 pageImage.flush();
30             readerScreenDirty = true;
31             readerPageImage.setElementAt(null, readerPageNum - 1);
32             setReaderRect();
33             clearRowElements();
34             repaintReaderScreen();
35             this.repaint();
36
37             return false;
38         }
39         else
40             return true;
41     }
42
43     public int indent(boolean right)
44     {
45         //
46         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
47 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").indent("+right+"")
48 );
49
50         if (right)
51         {
52             if (rightIndent)
53             {
54                 return 2*readerEnWidth;
55             }
56         }
57         else
58         {
59             if (unorderedList)
60             {
61                 int tabs = listAccount.size() > 1?listAccount.size() - 1:0;
62                 return 2*(tabs + (termDefinition?1:0))*readerEnWidth;
63             }
64             if (orderedList)
65             {
66                 return 2*listAccount.size()*readerEnWidth;
67             }
68             if (leftIndent)
69             {
70                 return 2*readerEnWidth;

```

```

1      }
2      }
3
4      return 0;
5  }
6
7  public Insets insets()
8  {
9      // dbg("_p_tb_db"+"theTableNumber+").insets());
10
11      return insets;
12  }
13
14  public final void loadPageData(int pageIndex)
15  {
16      //
17      dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
18  aderContent.length())<30?ZZ.substring(readerContent.length():MT)+").loadPageData("+pag
19  eIndex+"): readerPageStart.size() = "+readerPageStart.size()+", readerEndIndex =
20  "+readerEndIndex+", lineImages.size() = "+lineImages.size()+"; listAccount.size() =
21  "+listAccount.size());
22
23      Image pageImage = (Image) readerPageImage.elementAt(pageIndex);
24      listAccount = (Vector) readerListAccts.elementAt(pageIndex);
25      linkHeads = (Vector) readerLinkHeads.elementAt(pageIndex);
26      linkRects = (Vector) readerLinkRects.elementAt(pageIndex);
27      zoomHeads = (Vector) readerZoomHeads.elementAt(pageIndex);
28      zoomRects = (Vector) readerZoomRects.elementAt(pageIndex);
29      pageComps = (Vector) readerPageComps.elementAt(pageIndex);
30      readerStartIndex = ((Integer)
31  readerPageStart.elementAt(pageIndex)).intValue();
32
33      hyperHead = (URL) readerLinkStart.elementAt(pageIndex);
34      this.restoreStyle ((Integer) readerTypeStart.elementAt(pageIndex));
35      this.setFont ((Font) readerFontStart.elementAt(pageIndex)
36      , (Color) readerTintStart.elementAt(pageIndex));
37
38      if (pageImage != null)
39          readerScreen = pageImage;
40  }
41
42  public final Dimension minimumSize()
43  {
44      //
45      dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
46  aderContent.length())<30?ZZ.substring(readerContent.length():MT)+").minimumSize("+mini
47  mumSize.width+", "+minimumSize.height+")");
48
49      return minimumSize;
50  }
51
52  public boolean mouseDown(Event evt, int x, int y)
53  {
54      //
55      dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
56  aderContent.length())<30?ZZ.substring(readerContent.length():MT)+").mouseDown("+x+", "+
57  y+")");
58
59      if
60      ( !readerFrame
61      && zoomable
62      && readerContent.length() > 0
63      )
64      {
65          Component ancestor = getParent();
66          if (ancestor != null)
67          {
68              if (ancestor instanceof ez_html)
69              {
70                  return false;

```

```

1      )
2      else
3          if (ancestor instanceof _p_tc_db)
4          {
5              while (ancestor != null)
6              {
7                  if (ancestor instanceof _f_tb_db &&
8                      ((_f_tb_db)ancestor).justDragging)
9                  {
10                     return false;
11                 }
12                 ancestor = ancestor.getParent();
13             }
14         }
15     }
16     ez_HTML.showStatus(ZN);
17 }
18 else
19 for (int i = 0; i < zoomRects.size(); i++)
20 {
21     if (zoomRects.elementAt(i) instanceof Rectangle
22         && ((Rectangle)zoomRects.elementAt(i)).inside(x,y))
23     {
24         if (zoomHeads.elementAt(i) instanceof String)
25         {
26             ez_HTML.showStatus(ZI);
27         }
28         return true;
29     }
30 }
31
32 return super.mouseDown(evt, x, y);
33 }
34
35 public boolean mouseMove(Event evt, int x, int y)
36 {
37     for (int i = 0; i < linkRects.size(); i++)
38     if (linkRects.elementAt(i) instanceof Rectangle
39         && linkHeads.elementAt(i) instanceof URL
40         && ((Rectangle)linkRects.elementAt(i)).inside(x,y))
41     {
42         ez_HTML.showStatus(((URL)linkHeads.elementAt(i)).toString());
43         return true;
44     }
45
46     for (int i = 0; i < zoomRects.size(); i++)
47     if (zoomRects.elementAt(i) instanceof Rectangle
48         && zoomHeads.elementAt(i) instanceof String
49         && ((Rectangle)zoomRects.elementAt(i)).inside(x,y))
50     {
51         ez_HTML.showStatus(ZM);
52         return true;
53     }
54
55     if
56     (   zoomable
57     && readerContent.length() > 0
58     )
59     {
60         ez_HTML.showStatus(ez_HTML.ZM);
61         return true;
62     }
63     ez_HTML.showStatus(_p_rs_db.MT);
64
65     return super.mouseMove(evt, x, y);
66 }
67
68 public boolean mouseUp(Event evt, int x, int y)
69 {

```



```

1      //
2      dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
3      aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+"").mouseUp() ");
4
5      for (int i = 0; i < linkRects.size(); i++)
6      {
7          if (linkRects.elementAt(i) instanceof Rectangle
8          && linkHeads.elementAt(i) instanceof URL
9          && ((Rectangle)linkRects.elementAt(i)).inside(x,y))
10         {
11             ez_HTML.getAppletContext().showDocument
12             ( (URL)linkHeads.elementAt(i)
13             , "_blank" // opens new page in (new, if
14             cooperative) browser window
15             );
16             return true;
17         }
18     }
19
20     if
21     ( !readerFrame
22     && zoomable
23     && readerContent.length() > 0
24     )
25     {
26         String title = null;
27         Component ancestor = getParent();
28         if (ancestor != null)
29         {
30             if (ancestor instanceof ez_html)
31             {
32                 return super.mouseDown(evt, x, y);
33             }
34             else
35             if (ancestor instanceof _p_tc_db)
36             {
37                 boolean findTitle = true;
38                 while (ancestor != null)
39                 {
40                     if (ancestor instanceof _p_tb_db && findTitle)
41                     {
42                         title = ((_p_tb_db)ancestor).theCaption;
43                         findTitle = false;
44                     }
45                     if (ancestor instanceof _f_tb_db &&
46                     ((_f_tb_db)ancestor).justDragging)
47                     {
48                         return super.mouseDown(evt, x, y);
49                     }
50                     ancestor = ancestor.getParent();
51                 }
52             }
53         }
54         // dbg("... making new _f_rs_db");
55
56         ez_HTML.showStatus(ZW);
57         _f_rs_db panelFrame = new _f_rs_db
58         ( title
59         , readerContent
60         , ez_HTML
61         , 15 // max columns
62         , true // present markers
63         , true // present footer
64         );
65         return true;
66     }
67     else
68     for (int i = 0; i < zoomRects.size(); i++)
69     {
70         if (zoomRects.elementAt(i) instanceof Rectangle

```

```

1      && ((Rectangle)zoomRechts.elementAt(i)).inside(x,y))
2      {
3          if (zoomHeads.elementAt(i) instanceof String)
4          {
5              ez_HTML.showStatus(ZX);
6              _o_im_db.zoomImage((String)zoomHeads.elementAt(i), ez_HTML);
7          }
8          return true;
9      }
10     }
11
12     return super.mouseUp(evt, x, y);
13 }
14
15     public final void nextColumn()
16     {
17         //
18         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
19 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").nextColumn(): old
20 colNum = "+colNum);
21
22         colNum++;
23         lastColumn = colNum == numColumns - 1;
24
25         yb = readerRect.y;
26     line
27         xb += colSpacing;
28     column area
29         te += colSpacing;
30         xe += colSpacing;
31     column area
32     }
33
34     public final void paint(Graphics g)
35     {
36         //
37         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
38 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").paint(g):
39 isEnabled = "+isEnabled()+", isShowing = "+isShowing()+", isValid = "+isValid()+",
40 isVisible = "+isVisible());
41
42         if (readerScreenDirty)
43         {
44             if (readerScreen != null)
45                 readerScreen.flush();
46
47             readerScreen = createImage(this.size().width, this.size().height);
48
49             Graphics q = this.readerScreen.getGraphics();
50             paintScreen(q); // adds components (paintArea) for this page & saves
51 then
52             q.dispose();
53
54             readerScreenDirty = false;
55             readerPageImage.setElementAt(readerScreen, readerPageNum - 1);
56             this.invalidate();
57             this.validate();
58         }
59         else if (this.countComponents() != pageComps.size()) // restore components
60         {
61             //
62             dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
63 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").paint(g):
64 removeAll() ");
65             this.removeAll();
66             for (int i = 0; i < pageComps.size(); i++)
67             {
68                 _p_tb_db table = (_p_tb_db)pageComps.elementAt(i);
69                 //
70                 dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re

```

```

1  aderContent.length() < 30 ? ZZ.substring(readerContent.length() : MT) : "" ).paint(g) : adding
2  table "+table.theTableNumber);
3
4      this.add(table);
5      // table.show(); // just in case it was hidden
6  }
7  }
8
9      if (g != null)
10     {
11         g.drawImage(readerScreen, 0, 0, this);
12         // this.validate();
13     }
14 }
15
16     private final int paintArea // returns string
17     index of first non-drawn character
18     ( Graphics g
19     ,   int beginIndex
20     ,   String textContent
21     ,   int numCols
22     )
23     {
24         //
25         dbg("_p_rs_db"+"readerContent.substring(0, Math.min(30, readerContent.length()))+(re
26 aderContent.length() < 30 ? ZZ.substring(readerContent.length() : MT) : "" ).paintArea(beginInd
27 ex = "+beginIndex+"");
28         // long time = System.currentTimeMillis();
29
30         int endText = textContent.length();
31         endIndex = beginIndex;
32
33         numColumns = numCols;
34         colNum = 0;
35         int colMargin = 2*readerEnWidth; // 2 n-spaces between
36 cols
37         colWidth = (readerRect.width - (numColumns-1)*colMargin)/numColumns;
38         colSpacing = colWidth + colMargin;
39
40         String nextWord = MT;
41
42         xb = readerRect.x; // local left edge of
43 the column area
44         te = xb + indent(false); // end position of
45 this line's words & graphics
46         xe = xb + colWidth; // local right edge
47 of the column area
48
49         nextWidth = 0;
50
51         yb = readerRect.y; // local starting top
52 of first line
53         ye = readerRect.y + readerRect.height; // local maximum
54 bottom of last line
55         // dbg("... yb = "+yb+"(initially), ye = "+ye);
56
57         lastColumn = colNum == numColumns - 1;
58         lastRow = false;
59         controlSize = textHeight*2/3;
60
61         if (markings)
62             setMarkers(g, controlSize);
63
64         clearRowElements();
65
66         StringTokenizer tagBlocks = new StringTokenizer // break up the text
67 into normal tagBlocks
68         (textContent.substring(beginIndex), PF, true); // final true returns
69 token on next call
70

```

```

1      draw: while (tagBlocks.hasMoreTokens() && !(yb > ye && lastColumn))
2      {
3      follow
4          String tagBlock = tagBlocks.nextTok(PF);
5      text>body text],..., [<tag text>body text]
6
7          // handle all html tag instructions
8
9          if (!plaintext && tagBlock.equals(PF))
10 tag prefix?
11          {
12              String tagText = tagBlocks.nextTok(SF);
13
14              if (tagBlocks.hasMoreTokens())
15                  tagBlock = tagBlocks.nextTok();
16 tag suffix
17              else
18                  tagBlock = MT;
19
20              if (tagBlock.equals(SF))
21              {
22                  int endIncr = addTag(tagText, g);
23 paintRow() indexing
24                  if (endIncr < 0)
25                  {
26                      endIndex -= endIncr;
27 was actually done
28                      break draw;
29 management (no room for it)
30                  }
31                  else
32                  {
33                      endIndex += endIncr;
34 that, what's next
35                  }
36                  tagBlock = MT;
37 transfer text capture
38              }
39              else
40 syntax error
41              {
42                  tagBlock = PF + tagText + tagBlock;
43              }
44          }
45
46          // handle all html text content
47
48          if (!tagBlock.equals(PF) && !tagBlock.equals(MT))
49          {
50              if (!addBlock(tagBlock,g,true)) break draw;
51          }
52          } // end of draw
53          endIndex += paintRow(g, 0);
54 pending line elements
55
56          while (endIndex < endText
57              && textContent.substring(endIndex++).startsWith(BL));
58
59          savePageData(readerPageNum, endIndex);
60
61          estimateNumPages();
62          if (markings)
63              paintMarkers(g, controlSize);
64
65          // dbg("area paint duration = "+(System.currentTimeMillis()-time)+" ms.");
66
67          return endIndex;
68      }
69
70      public final void paintMarkers(Graphics g, int controlSize)

```

```

1      {
2          if (g == null) return;
3
4          g.setColor(getForeground());
5
6          if (!readerFrame)                                // right arrow
7          {
8              int x = aheadRect.x + aheadRect.width - 2
9              , y = aheadRect.y + aheadRect.height/2;
10             for (int arrow = 0; arrow < controlSize; arrow++)
11             {
12                 g.drawLine(x - arrow,y + arrow/2,x - arrow,y - arrow/2);
13             }
14         }
15     }
16
17     public final synchronized int paintRow(Graphics g, int retain)
18     {
19         //
20         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
21 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").paintRow(g,
22 "+retainCount+") @ ("+"xb"+"+"yb+")", left indent = "+indent(false)+", right indent =
23 "+indent(true)+"; stack("+dataValues.size()+")");
24
25         int retainCount = Math.max(0,retain);           // ensure reasonable results
26         int maxAscent = 0;                               // measure the max baseline
27         ascent for output
28         int maxDescent = 0;                             // measure the max baseline
29         descent for output
30         int lineWidth = 0;                              // measure the full width of this
31         output
32         int indexIncr = 0;                              // measure the number of
33         characters represented by this line
34
35         for (int i = 0; i < dataValues.size() - retainCount; i++)
36         {
37             Rectangle d = (Rectangle)dataValues.elementAt(i);
38
39             maxAscent = Math.max(maxAscent, d.y);
40             maxDescent = Math.max(maxDescent, d.height);
41             lineWidth += d.width;
42         }
43         yb += maxAscent + maxDescent;
44
45         while (yb > ye)                                  // start a new column?
46         {
47             dbg("... yb > ye");
48             if (lastColumn) return indexIncr;           // no more columns left to fill
49             nextColumn();
50             yb += maxAscent + maxDescent;
51         }
52
53         int itsStyle = LEFT;
54         if (styleCodes.size() > 0)
55         {
56             if
57             ( g != null
58             && wordColors.size() > 0
59             && charsFonts.size() > 0
60             )
61             {
62                 g.setColor((Color)wordColors.firstElement());
63                 g.setFont((Font)charsFonts.firstElement());
64             }
65
66             restoreStyle(itsStyle = ((Integer)styleCodes.firstElement()).intValue());
67         }
68
69         int x = xb + indent(false) + textAlign*(effectiveWidth() - lineWidth)/2;

```



```

1      )
2      if (orderedList)
3      {
4          _o_li_db.drawLabel(this, x, y, g);
5      }
6  )
7
8      boolean doUnderline = (itsStyle & UNDERLINE) != 0 || (itsStyle &
9  HYPERLINK) != 0;
10     boolean doLineStrike = (itsStyle & STRIKETHRU) != 0;
11
12     int shift = 0;
13
14     if ((itsStyle & SUBSCRIPT) != 0)
15         shift += (textHeight+3)/4 - 1;
16     if ((itsStyle & SUPSCRIPT) != 0)
17         shift -= (textHeight+3)/4 - 1;
18
19     if (g != null)
20     {
21         if (doUnderline)
22             g.drawLine(x, y + 2 + shift, x + d.width, y + 2 + shift);
23         if (doLineStrike)
24             g.drawLine(x, y - (textHeight+3)/4 + shift, x + d.width, y -
25 (textHeight+3)/4 + shift);
26
27         g.drawString(wordBlock, x, y + shift);
28     }
29
30     if
31     (   hyperLinks.size() > 0
32     && hyperLinks.firstElement() != null
33     )
34     {
35         // if (debug && g != null) g.drawRect(x, y - d.y + shift, d.width +
36 readerEnWidth/2, d.y + d.height);
37         linkHeads.addElement(hyperLinks.firstElement());
38         linkRects.addElement(new Rectangle(x, y - d.y + shift, d.width +
39 readerEnWidth/2, d.y + d.height));
40     }
41 }
42
43 x += d.width;
44 indexIncr += d.x;
45
46 if
47 (   lineImages.size() > 0
48 && dataValues.size() > 0
49 && wordBlocks.size() > 0
50 && hyperLinks.size() > 0
51 && styleCodes.size() > 0
52 && wordColors.size() > 0
53 && charsFonts.size() > 0
54 )
55 {
56     lineImages.removeElementAt(0);
57     dataValues.removeElementAt(0);
58     wordBlocks.removeElementAt(0);
59     hyperLinks.removeElementAt(0);
60     styleCodes.removeElementAt(0);
61     wordColors.removeElementAt(0);
62     charsFonts.removeElementAt(0);
63 }
64 else
65 {
66     /*   dbg
67     (   "unexpected size(s) in paintRow("+retain+"):\n"
68     +   "lineImages.size() " + lineImages.size() + "\n"
69     +   "dataValues.size() " + dataValues.size() + "\n"
70     +   "wordBlocks.size() " + wordBlocks.size() + "\n"

```

```

1      + "hyperLinks.size() " + hyperLinks.size() + "\n"
2      + "styleCodes.size() " + styleCodes.size() + "\n"
3      + "wordColors.size() " + wordColors.size() + "\n"
4      + "charsFonts.size() " + charsFonts.size() + "\n"
5      );
6      */
7      }
8      maxWidth = Math.max(maxWidth, lineWidth);
9      maxHeight = Math.max(maxHeight, yb - readerRect.y);
10
11      savePageData(readerPageNum, endIndex + indexIncr);      // for the next page
12
13      // dbg("... final stack("+dataValues.size()+")");
14      return indexIncr;
15  }
16
17  private final synchronized void paintScreen(int width)      // test (null) paint
18  for sizing
19  {
20      // dbg("paintScreen("+width+")");
21
22      Dimension s = getToolkit().getScreenSize();
23      int wide = width + insets.left + insets.right + 2;      // one column wide
24      int tall = s.height;      // super column
25
26      height
27      Rectangle r = new Rectangle
28      (
29      0
30      ,
31      0
32      ,
33      Math.min(wide, s.width)      // set to indicated
34      ,
35      tall
36      );
37      paintScreen(null, r);
38  }
39
40  private final synchronized void paintScreen(Graphics g)
41  {
42      //
43      dbg("_p_rs_db("+readerContent.substring(0,Math.min(30,readerContent.length()))+(re
44      aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").paintScreen(g):
45      getBackground() = "+getBackground());
46
47      Rectangle r = bounds();      // set to full area
48      (this.bounds())
49      paintScreen(g, r);
50  }
51
52  private final synchronized void paintScreen(Graphics g, Rectangle r)
53  {
54      //
55      dbg("_p_rs_db("+readerContent.substring(0,Math.min(30,readerContent.length()))+(re
56      aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").paintScreen(g):
57      getBackground() = "+getBackground());
58
59      setReaderRect(r);      // set to full area
60      (this.bounds())
61
62      if (g != null)
63      {
64          g.setColor(getBackground());
65          g.fillRect(0, 0, r.width, r.height);
66          for (int i = 0; i < border; i++)
67          {
68              g.draw3DRect
69              (
70              i + spacing/2
71              ,
72              i + spacing/2
73              ,
74              r.width - 1 - spacing - 2*i
75              ,
76              r.height - 1 - spacing - 2*i
77              ,
78              false
79              );
80          }
81      }

```



```

1      // background fill, if requested
2
3      Image background = ez_HTML.getBackImage();
4      if (background != null)
5      {
6          Point p = this.location();
7          int w = background.getWidth(this);
8          int h = background.getHeight(this);
9          if (w > 0 && h > 0)
10         {
11             int x = this.size().width;
12             int y = this.size().height;
13             for (int i = 0; i < x; i += w)
14             {
15                 for (int j = 0; j < y; j += h)
16                 {
17                     g.drawImage(background, i-p.x, j-p.y, this);
18                 }
19             }
20         }
21     }
22
23 }
24
25 clearPage();
26 maxHeight = 0;
27
28 int numCols = calculateReaderColumns();
29 if (numCols != readerNumCols)
30 {
31     readerNumCols = numCols;
32     resetReaderView();
33 }
34
35 if (readerFrame)
36     readerLeading = 2;
37 else
38     readerLeading = 0;
39
40 // paint the content area
41
42 loadPageData(readerPageNum-1);
43 markings = readerMarker && !readerFooter;
44 readerEndIndex = paintArea
45 (
46     g
47     ,   readerStartIndex
48     ,   readerContent
49     ,   readerNumCols
50 );
51 readerStartIndex = readerEndIndex - 1;
52 estimateNumPages();
53
54 if (preferredSize.width < 0
55     || preferredSize.height < 0)
56 {
57     Dimension screenSize = getToolkit().getScreenSize();
58
59     preferredSize.width = Math.min
60     (
61         screenSize.width
62         ,   Math.max
63         (
64             minimumSize.width
65             ,   insets.left + insets.right + 2
66             +   maxWidth
67         )
68     );
69
70     preferredSize.height = Math.min
71     (
72         screenSize.height
73         ,   Math.max
74         (
75             minimumSize.height

```

```

1          insets.top + insets.bottom + 2
2          + maxHeight * (colNum + 1)
3          + (readerFooter?readerFontSize[readerFootIndex]*3/2:0)
4      )
5      );
6      //
7      dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
8      aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").paintScreen(): new
9      preferredSize("+preferredSize.width+","+preferredSize.height+"); complete =
10     "+(readerEndIndex == readerTextLength)+"; maxHeight = "+maxHeight+", readerNumCols =
11     "+readerNumCols);
12     }
13     }
14
15     public final Dimension preferredSize()
16     {
17         return preferredSize(standardLineWidth());
18     }
19
20     public final Dimension preferredSize(int prefLineWidth)
21     {
22         if (preferredSize.width < 0 || preferredSize.height < 0)
23         {
24             resetReaderView();
25             paintScreen(prefLineWidth);
26             resetReaderView();
27             //
28             dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
29             aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").preferredSize("+pr
30             eferredSize.width+","+preferredSize.height+") - from null screen layout");
31
32             return preferredSize;
33         }
34         else
35         {
36             //
37             dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
38             aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").preferredSize("+pr
39             eferredSize.width+","+preferredSize.height+") - prior layout");
40
41             return preferredSize;
42         }
43     }
44
45     public synchronized void removeAll()    // override, because some browsers fail
46     here w/o layout manager
47     {
48         super.removeAll();
49
50         int n = countComponents();
51         if (n > 0)
52             this.invalidate();
53         while (n > 0)
54         {
55             Component c = getComponent(n-1);
56             c.hide();
57             c.invalidate();
58             remove(c);
59             n = countComponents();
60         }
61     }
62
63     public final void repaintReaderScreen()
64     {
65         //
66         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
67         aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").repaintReaderScree
68         n()");
69
70         if (readerPageImage.elementAt(readerPageNum - 1) == null)

```

```

1      {
2          readerScreenDirty = true;
3      }
4      this.repaint();
5  }
6
7      public final synchronized void resetReaderView()
8      {
9          //
10         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
11 aderContent.length())<30?ZZ.substring(readerContent.length()):MT)+").resetReaderView()"
12 );
13
14         readerStartIndex    = 0;
15         readerEndIndex      = 1;
16         readerNumPages      = 0;
17         readerEstPages      = 1;
18         readerPageNum       = 1;
19
20         readerPageImage.removeAllElements();
21         readerListAccts.removeAllElements();
22         readerLinkHeads.removeAllElements();
23         readerLinkRects.removeAllElements();
24         readerZoomHeads.removeAllElements();
25         readerZoomRects.removeAllElements();
26         readerPageComps.removeAllElements();
27         readerPageStart.removeAllElements();
28
29         readerLinkStart.removeAllElements();
30         readerTypeStart.removeAllElements();
31         readerFontStart.removeAllElements();
32         readerTintStart.removeAllElements();
33
34         clearPage();
35         maxWidth = 0;
36         newListItem = false;
37         plaintext = false;
38
39         readerScreenDirty = true;
40
41         restoreStyle(0);
42         this.setFont(readerScreenFont, getForeground());
43
44         savePageData(0, readerEndIndex);
45     }
46
47     public void reshape(int x, int y, int width, int height)
48     {
49         super.reshape(x, y, width, height);
50         resetReaderView();
51
52         //
53         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
54 aderContent.length())<30?ZZ.substring(readerContent.length()):MT)+").reshape("+x+",
55 "+y+", "+width+", "+height+")");
56
57         repaint();
58     }
59
60     public final void restoreStyle(Integer theStyleCode)
61     {
62         restoreStyle(theStyleCode.intValue());
63     }
64
65     public final void restoreStyle(int itsStyle)
66     {
67         textAlign      = (itsStyle & POSITION    );
68         underlining    = (itsStyle & UNDERLINE ) != 0;
69         strikethrough  = (itsStyle & STRIKE THRU ) != 0;

```

```

1      subscript      = (itsStyle & SUBSCRIPT ) != 0 && (itsStyle & SUPSCRIPT ) ==
2      0;
3      superscript    = (itsStyle & SUPSCRIPT ) != 0 && (itsStyle & SUBSCRIPT ) ==
4      0;
5      hyperlink      = (itsStyle & HYPERLINK ) != 0;
6      leftIndent     = (itsStyle & LEFTINDENT ) != 0;
7      rightIndent    = (itsStyle & RIGHTINDENT) != 0;
8      hidingText     = (itsStyle & HIDDENTEXT ) != 0;
9      unorderedList  = (itsStyle & UNORDERLIST) != 0;
10     orderedList    = (itsStyle & ORDEREDLIST) != 0;
11 }
12
13     private final void savePageData(int nextPageIndex, int currEndIndex)
14     {
15         //
16         dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
17 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").savePageData("+nex
18 tPageIndex+","+currEndIndex+"): new = "+(readerNumPages == 0 && nextPageIndex ==
19 readerPageStart.size()+", readerEndIndex = "+readerEndIndex+", lineImages.size() =
20 "+lineImages.size()+"; listAccount.size() = "+listAccount.size());
21
22         if (readerNumPages == 0
23         && nextPageIndex == readerPageStart.size())    // the first time reaching
24 this page
25         {
26             readerPageImage.addElement(null);          // readerScreen is loaded by
27 paint(g)
28
29             Vector currList = new Vector(listAccount.size());
30             for (int i = 0; i < listAccount.size(); i++)
31             {
32                 Point p = (Point)listAccount.elementAt(i);
33                 currList.addElement(new Point(p.x, p.y));
34             }
35             listAccount = currList;
36             readerListAccts.addElement(listAccount);
37             readerLinkHeads.addElement(new Vector());
38             readerLinkRects.addElement(new Vector());
39             readerZoomHeads.addElement(new Vector());
40             readerZoomRects.addElement(new Vector());
41             readerPageComps.addElement(new Vector());    // components are added by
42 placeTable(g)
43             readerPageStart.addElement(new Integer(currEndIndex - 1));
44
45             if (nextPageIndex > 0
46             && lineImages.size() > 0)                    // unpainted elements in
47 stack
48             {
49                 readerLinkStart.addElement((URL)hyperLinks.firstElement());
50                 readerFontStart.addElement((Font)charsFonts.firstElement());
51                 readerTintStart.addElement((Color)wordColors.firstElement());
52                 readerTypeStart.addElement((Integer)styleCodes.firstElement());
53             }
54             else
55             {
56                 readerLinkStart.addElement(hyperHead);
57                 readerFontStart.addElement(textFont);
58                 readerTintStart.addElement(textColor);
59                 readerTypeStart.addElement(styleCode());
60             }
61         }
62         else
63             // update next page initial
64             state vectors
65             {
66                 readerPageStart.setElementAt(new Integer(currEndIndex - 1),
67                 nextPageIndex);
68                 if (nextPageIndex > 0
69                 && lineImages.size() > 0)                // unpainted elements in
70 stack
71                 {

```

```

1      readerLinkStart.setElementAt((URL)hyperLinks.firstElement(),
2  nextPageIndex);
3      readerFontStart.setElementAt((Font)charsFonts.firstElement(),
4  nextPageIndex);
5      readerTintStart.setElementAt((Color)wordColors.firstElement(),
6  nextPageIndex);
7      readerTypeStart.setElementAt((Integer)styleCodes.firstElement(),
8  nextPageIndex);
9      }
10     else
11     {
12         readerLinkStart.setElementAt(hyperHead, nextPageIndex);
13         readerFontStart.setElementAt(textFont, nextPageIndex);
14         readerTintStart.setElementAt(textColor, nextPageIndex);
15         readerTypeStart.setElementAt(styleCode(), nextPageIndex);
16     }
17 }
18 }
19
20 public void setAllFonts()
21 {
22     readerFooterFont = new Font(FS, Font.PLAIN, readerFontSize[readerFootIndex]);
23     readerScreenFont = new Font(FS, Font.PLAIN, readerFontSize[readerSizeIndex]);
24
25     footerTextHeight = readerFontSize[readerFootIndex]*5/4;
26
27     setFont(readerScreenFont, getForeground());
28     readerEnWidth = getFontMetrics(readerScreenFont).stringWidth("n");
29 }
30
31 public final void setFont(Font f, Color c)
32 {
33     textColor = c;
34     this.setFont(f);
35 }
36
37 public final void setFont(Font f)
38 {
39     textFont = f;
40     textMetrics = getFontMetrics(f);
41     textHeight = f.getSize()*5/4;
42     textHalfSpace = (textHeight + readerLeading)/2;
43
44     //
45     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
46 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").setFont():
47 textHeight = "+textHeight);
48 }
49
50 public synchronized final void setFontSizeIndex(int newSizeIndex)
51 {
52     //
53     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
54 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").setFontSize("+newS
55 izeIndex+": # comps = "+countComponents());
56
57     readerSizeIndex = newSizeIndex;
58     readerFootIndex = readerSizeIndex - 2;
59
60     readerSizeIndex = Math.max(Math.min(readerSizeIndex, readerFontSize.length -
61 1), 0);
62     readerFootIndex = Math.max(Math.min(readerFootIndex, readerFontSize.length -
63 1), 0);
64
65     setAllFonts();
66 }
67
68 public void setMarkers(Graphics g, int controlSize)
69 {
70     aheadRect.reshape

```

```

1      ( readerRect.x + readerRect.width - 2 - (controlSize+4)*1
2      ,   readerRect.y + readerRect.height - 4 - controlSize
3      ,   controlSize + 4
4      ,   controlSize + 4
5      );
6  }
7
8  public void setReaderRect()
9  {
10     //
11     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
12 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").setReaderRect()");
13
14     setReaderRect(this.bounds());
15 }
16
17 public void setReaderRect(Rectangle r)
18 {
19     readerRect.reshape
20     ( insets.left
21     , insets.top
22     , r.width - 1 - insets.left - insets.right
23     , r.height - 1 - insets.top - insets.bottom
24     - (readerFooter?readerFontSize[readerFootIndex]*3/2:0)
25     );
26     firstWord = true;
27
28     //
29     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
30 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").setReaderRect"+"r.
31 x"+"r.y"+"r.width"+"r.height+"):
32 readerRect.reshape"+"readerRect.x"+"readerRect.y"+"readerRect.width"+"readerRect
33 .height+");
34 }
35
36 public final void setZoomable(boolean zoomable)
37 {
38     this.zoomable = zoomable;
39 }
40
41 public final int standardLineWidth()
42 {
43     int numChar = 30;           // the control parameter: min # of n's per line
44 of text
45     int stndLineWidth = numChar * readerEnWidth;
46
47     //
48     dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
49 aderContent.length()<30?ZZ.substring(readerContent.length():MT)+").standardLineWidth(
50 ) = "+stndLineWidth);
51
52     return stndLineWidth;
53 }
54
55 public final boolean startNewParagraph(int halfSpaces)
56 {
57     if (yb > readerRect.y)           // unless we're at the top
58     {
59         yb += halfSpaces * textHalfSpace; // drop down to next baseline
60         if (yb > ye)
61         {                               // not enough room in this (partial?)
62 column
63             if (lastColumn) return false; // no more columns left to fill
64             nextColumn();
65         }
66     }
67     te = xb + indent(false);           // start a new line
68     maxHeight = Math.max(maxHeight, yb - readerRect.y);
69
70     return true;

```

```

1      }
2
3      public void stepFontSize(boolean smaller)
4      {
5          //
6          dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
7 aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").stepFontSize(small
8 er = "+smaller+"");
9
10         if (smaller)    readerSizeIndex--;
11         else            readerSizeIndex++;
12
13         setFontSizeIndex(readerSizeIndex);
14     }
15
16     public final Integer styleCode()
17     {
18         return new Integer
19         (   textAlign
20           + ((underlining)      ? UNDERLINE   :0)
21           + ((strikethrough)    ? STRIKETHRU   :0)
22           + ((subscript && !supscript) ? SUBSCRIPT :0)
23           + ((supscript && !subscript) ? SUPSCRIPT :0)
24           + ((hyperLink)       ? HYPERLINK    :0)
25           + ((leftIndent)      ? LEFTINDENT   :0)
26           + ((rightIndent)     ? RIGHTINDENT  :0)
27           + ((hidingText)      ? HIDDENTEXT   :0)
28           + ((unorderedList)   ? UNORDERLIST  :0)
29           + ((orderedList)     ? ORDEREDLIST  :0));
30     }
31
32     public void update(Graphics g)
33     {
34         // fillRect is NOT performed here to eliminate blank screen boredom during
35         offscreen drawing
36
37         // g.setColor(getBackground());
38         // g.fillRect(0, 0, width, height);
39         // g.setColor(getForeground());
40
41         paint(g);
42     }
43
44     public _p_rs_db
45     (   String readerContent
46       ,   ez_html ez_HTML
47       ,   _f_rs_db frame
48       ,   boolean readerFrame
49       ,   int maxColumns
50       ,   boolean marker
51       ,   boolean footer
52       ,   boolean zoomable
53     )
54     {
55         this
56         (   readerContent
57         ,   ez_HTML
58         ,   frame
59         ,   readerFrame
60         ,   maxColumns
61         ,   marker
62         ,   footer
63         ,   readerFrame?new Color(239,239,239):ez_HTML.getBackColor()
64         ,   readerFrame?Color.black:ez_HTML.getForeColor()
65         ,   0           // no border
66         ,   10          // no padding
67         ,   0           // no spacing
68         ,   zoomable    // zoomability
69         );
70     }

```

```

1
2     public _p_rs_db
3     (   String readerContent
4         ,   ez_html ez_HTML
5         ,   _f_rs_db frame
6         ,   boolean readerFrame
7         ,   int maxColumns
8         ,   boolean marker
9         ,   boolean footer
10        ,   Color panelBg
11        ,   Color panelFg
12        ,   int border
13        ,   int padding
14        ,   int spacing
15        ,   boolean zoomable
16    )
17    {
18        //
19        dbg("_p_rs_db"+"readerContent.substring(0,Math.min(30,readerContent.length()))+(re
20aderContent.length()<30?ZZ.substring(readerContent.length()):MT)+").<init>\n"+readerCo
21ntent);
22
23        this.setLayout(null);
24
25        this.ez_HTML = ez_HTML;
26        this.frame = frame;
27        this.readerFrame = readerFrame;
28        this.readerHeader = ez_HTML.getTitle() != null;
29        this.readerContent = readerContent;
30        this.readerTextLength = readerContent.length();
31        this.charEntConverter = new _o_nt_db();
32        this.maxColumns = Math.max(1, maxColumns);
33        this.readerMarker = marker;
34        this.readerFooter = footer;
35        setBackground(panelBg);
36        setForeground(panelFg);
37        this.border = border;
38        this.padding = padding;
39        this.spacing = spacing;
40        this.zoomable = zoomable;
41
42        insets.top      =
43        insets.left     = border + padding + spacing/2;
44        insets.bottom   =
45        insets.right    = border + padding + spacing - spacing/2;
46
47        setAllFonts();
48        if (readerFrame)
49        {
50            stepFontSize(false);
51            reader screens
52        }
53        readerEnWidth = getFontMetrics(readerScreenFont).stringWidth("n");
54    }
55 }
56
57 /**
58  *  %W% %E% Everett Stoub
59  *
60  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
61  *
62  *  a frame for presenting interactive multicolumn text and graphics
63  *
64  *  @author Everett Stoub %I%, %G%
65  */
66 import java.awt.*;
67 import java.applet.*;
68 import java.net.*;
69 import java.io.*;
70 import java.lang.*;

```



```

1  import java.util.*;
2
3  public final class _f_rs_db extends Frame implements Runnable
4  {
5      public ez_html      ez_HTML = null;
6
7      public p_rs_db      panel = null;
8      public p_pm_db      locator = null;
9      public Panel        control = null;
10     public Panel         subpanel = null;
11
12     public Thread        runner = null;
13     public int           messageNumber = 0;
14     public Label         messageLabel = null;
15     public boolean       stopped = false;
16     public String[]      message =
17     {
18         "About: EZ HTML Version 0.95a1..."
19         , "...Certified 100% Pure Java"
20         , "...More info: 888.IQN.JAVA"
21         , "...URL: www.ionsystems.com"
22         , "...Created by Everett Stoub"
23         , "...(c)'97, IQN Systems, Inc."
24     };
25
26     private String buttonName[] =
27     {
28         "Next"
29         , "Previous"
30         , "Font +"
31         , "Font -"
32         // , "Home"
33     };
34     public String CP = "Center";
35
36     public void start()
37     {
38         if (runner == null)
39         {
40             runner = new Thread(this);
41             runner.start();
42         }
43
44     public void stop()
45     {
46         if (runner != null && runner.isAlive())
47             runner.stop();
48         runner = null;
49
50     public void run()
51     {
52         while (runner != null)
53         {
54             if (messageNumber < message.length)
55                 messageLabel.setText(message[messageNumber++]);
56             try
57             {
58                 Thread.sleep(3000);
59             }
60             catch (InterruptedException e) {}
61         }
62
63     public boolean action(Event e, Object o)
64     {
65         if (e.target instanceof Button)
66         {
67             if (o == (buttonName[0]))    doViewerRight();      else
68             if (o == (buttonName[1]))    doViewerLeft();       else
69             if (o == (buttonName[2]))    doViewerSize(false);  else
70

```

```
1      if (o == (buttonName[3])) doViewerSize(true); /* else
2      if (o == (buttonName[4])) doViewerHome();      */
3
4      messageLabel.setText(message[messageNumber = 0]);
5  }
6
7      return false;
8  }
9
10     private final void dbg(String s)
11     {
12         if (debug) System.out.println(s);
13     }
14     private boolean debug = false;      // Print debugging info?
15
16     /* public final void doViewerHome()
17     {
18         // dbg("_f_rs_db.doViewerHome()");
19
20         if (panel.readerPageNum > 1)
21         {
22             subpanel.remove(panel);
23             panel.removeAll();
24             panel.readerPageNum = 1;
25             panel.loadPageData(panel.readerPageNum-1);
26             subpanel.add(CP, panel);
27         }
28         panel.repaintReaderScreen();
29         if (locator != null)
30         {
31             locator.repaint();
32         }
33     }
34
35     /* public final void doViewerLeft()
36     {
37         // dbg("_f_rs_db.doViewerLeft()");
38
39         if (panel.readerPageNum > 1)
40         {
41             subpanel.remove(panel);
42             panel.removeAll();
43             panel.readerPageNum--;
44             panel.loadPageData(panel.readerPageNum-1);
45             subpanel.add(CP, panel);
46         }
47         panel.repaintReaderScreen();
48         if (locator != null)
49         {
50             locator.repaint();
51         }
52     }
53
54     public final void doViewerRight()
55     {
56         // dbg("_f_rs_db.doViewerRight(): readerPageNum = "+panel.readerPageNum+",
57         readerEstPages = "+panel.readerEstPages);
58
59         if (panel.readerPageNum < panel.readerEstPages)
60         {
61             panel.readerPageNum++;
62         }
63         else if (panel.readerNumPages == 0)
64         {
65             panel.readerNumPages = panel.readerPageNum;
66         }
67         subpanel.remove(panel);
68         panel.removeAll();
69         panel.loadPageData(panel.readerPageNum-1);
70         panel.repaintReaderScreen();
```

```
1      subpanel.add(CP, panel);
2      if (locator != null)
3      {
4          locator.repaint();
5      }
6  }
7
8  public final void doViewerSize(boolean smaller)
9  {
10     //  dbg("_f_rs_db.doViewerSize(boolean "+smaller+"");
11
12     panel.doViewerSize(smaller);
13     subpanel.remove(panel);
14     panel.removeAll();
15     if (locator != null)
16     {
17         subpanel.remove(locator);
18         panel.resize
19         (   panel.size().width
20         ,   panel.size().height
21         +   locator.size().height
22         -   locator.preferredSize().height
23         );
24         locator.move
25         (   locator.location().x
26         ,   locator.location().y
27         +   locator.size().height
28         -   locator.preferredSize().height
29         );
30         locator.resize
31         (   locator.size().width
32         ,   locator.preferredSize().height
33         );
34         subpanel.add("South", locator);
35         locator.repaint();
36     }
37     subpanel.add(CP, panel);
38 }
39
40 public synchronized void dispose()
41 {
42     this.stop();
43
44     super.dispose();
45 }
46
47 public final boolean handleEvent(Event evt)
48 {
49     //  dbg("_f_rs_db.handleEvent("+evt+"");
50
51     switch (evt.id)
52     {
53         case Event.WINDOW_EXPOSE:      panel.requestFocus();      break;
54         case Event.WINDOW_DEICONIFY:    this.show();              break;
55         case Event.WINDOW_ICONIFY:      this.hide();              break;
56         case Event.WINDOW_DESTROY:      ez_HTML.disposeFrame(this); break;
57     }
58     return super.handleEvent(evt);
59 }
60
61 public final boolean keyDown(Event evt, int key)    // note: keyUp is not
62 utilized at all (missing in JDK 1.02 on Mac)
63 {
64     //  dbg("_f_rs_db.keyDown("+evt+", "+key+"");
65
66     boolean down = key == Event.DOWN;
67
68     switch (key)
69     {
70         //  case Event.HOME:      doViewerHome();      break;
```

```
1      case Event.PGDN:
2      case Event.RIGHT: doViewerRight(); break;
3      case Event.PGUP:
4      case Event.LEFT: doViewerLeft(); break;
5      case Event.UP:
6      case Event.DOWN: doViewerSize(down); break;
7
8      default: return false;
9  }
10  messageLabel.setText(message[messageNumber = 0]);
11
12  return true;
13  }
14
15  public final Dimension minimumSize()
16  {
17      Dimension s = getToolkit().getScreenSize();
18      if (debug)
19      {
20          s.width /= 2;
21          s.height /= 2;
22      }
23      else
24      {
25          s.height -= 50; // hack for menus
26      }
27
28      Dimension b = control.minimumSize();
29      Dimension r = subpanel.minimumSize();
30      Dimension f = new Dimension(Math.max(b.width, r.width), b.height + r.height);
31
32      double fArea = f.height * f.width;
33      double sArea = s.height * s.width;
34      if (fArea > sArea) // more than the screen area
35      {
36          f.width = s.width;
37          f.height = s.height;
38      }
39      else // adjust dimensions to
40  screen aspect
41      {
42          f.width = (int)Math.round(Math.sqrt(fArea * s.width / s.height));
43          f.height = (int)Math.round(Math.sqrt(fArea * s.height / s.width));
44      }
45      s.width = Math.min(s.width, f.width);
46      s.height = Math.min(s.height, f.height);
47
48      return s;
49  }
50
51  public boolean mouseDown(Event evt, int x, int y)
52  {
53      if (runner != null && runner.isAlive())
54      {
55          if (runner != null && runner.isAlive())
56          {
57              if (stopped)
58              {
59                  messageNumber = 0;
60                  runner.resume();
61              }
62              else
63              {
64                  runner.suspend();
65              }
66              stopped = !stopped;
67          }
68          else
69          {
70              stopped = false;
```

```

1         runner = new Thread(this);
2         runner.start();
3     }
4 }
5
6     return false;
7 }
8
9     public final Dimension preferredSize()
10    {
11        Dimension s = getToolkit().getScreenSize();
12        if (debug)
13        {
14            s.width /= 2;
15            s.height /= 2;
16        }
17        else
18        {
19            s.height -= 50; // hack for menus
20        }
21
22        Dimension b = control.preferredSize();
23        Dimension r = subpanel.preferredSize();
24        Dimension f = new Dimension(Math.max(b.width, r.width), b.height + r.height);
25
26        double fArea = f.height * f.width;
27        double sArea = s.height * s.width;
28        if (fArea > sArea) // more than the screen area
29        {
30            f.width = s.width;
31            f.height = s.height;
32        }
33        else // adjust dimensions to
34        screen aspect
35        {
36            f.width = (int)Math.round(Math.sqrt(fArea * s.width / s.height));
37            f.height = (int)Math.round(Math.sqrt(fArea * s.height / s.width));
38        }
39        s.width = Math.min(s.width, f.width);
40        s.height = Math.min(s.height, f.height);
41
42        return s;
43    }
44
45    public synchronized void reshape(int x, int y, int width, int height)
46    {
47        messageLabel.setText(message[messageNumber = 0]);
48        super.reshape(x, y, width, height);
49    }
50    public void update(Graphics g)
51    {
52        // g.setColor(getBackground());
53        // g.fillRect(0, 0, width, height);
54        // g.setColor(getForeground());
55
56        paint(g);
57    }
58
59    public _f_rs_db
60    (   String title
61        ,   String readerContent
62        ,   ez_html ez_HTML
63        ,   int maxColumns
64        ,   boolean marker
65        ,   boolean footer
66    )
67    {
68        super(title);
69
70        // dbg("_f_rs_db.<init>");

```

```

1
2     this.ez_HTML = ez_HTML;
3     this.setLayout(new BorderLayout());
4
5     subpanel = new Panel();
6     subpanel.setLayout(new BorderLayout());
7     this.add(CP, subpanel);
8
9     this.panel = new _p_rs_db
10    (   readerContent
11        ,   ez_HTML
12        ,   this
13        ,   true
14        ,   maxColumns
15        ,   false    // marker
16        ,   false    // footer
17        ,   false    // not zoomable
18    );
19    this.panel.insets.left  += 10;
20    this.panel.insets.right += 10;
21
22    subpanel.add(CP, panel);
23
24    if (footer)
25    {
26        locator = new _p_pm_db(panel, this);
27        subpanel.add("South", locator);
28    }
29
30    // control.add(new Label("ION EZ HTML v0.94a6 ")); //  [Δn-major].[Δn-minor][Δn-
31    tweak][Δa-fix]
32    control = new Panel();
33    control.setBackground(Color.gray);
34    control.setLayout(new FlowLayout(FlowLayout.LEFT));
35    this.add("South", control);
36
37    for (int i = 0; i < buttonName.length; i++)
38    {
39        Button b = new Button(buttonName[i]);
40        b.setBackground(Color.white);
41        control.add(b);
42    }
43    boolean mac = System.getProperty("os.name").equals("macos");
44    // control.add(new Label((mac?"@":"(c)")+" '97, ION Systems, Inc. 888.ION.JAVA"));
45    messageLabel = new Label(message[0]);
46    control.add(messageLabel);
47
48    this.pack();
49
50    // dbg("_f_rs_db.init<>: this.size() =
51    ("+this.size().width+", "+this.size().height+", panel.size() =
52    ("+panel.size().width+", "+panel.size().height+"));
53
54    this.show();
55    this.start();
56    }
57 }
58
59 /**
60  *  %W% %E% Everett Stoub
61  *
62  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
63  *
64  *  a simple panel to display page position and interactive markers
65  *
66  *  @author Everett Stoub %I%, %G%
67  */
68 import java.awt.*;
69
70 public final class _p_pm_db extends Panel

```

```

1  {
2      private _p_rs_db    panel = null;
3      private _f_rs_db    frame = null;
4      private int         controlSize = 8;
5
6      public Rectangle     aheadRect = new Rectangle();    // current page
7      public Rectangle     closeRect = new Rectangle();    // current page
8      public Rectangle     abackRect = new Rectangle();    // current page
9
10     private final void dbg(String s)
11     {
12         if (debug) System.out.println(s);
13     }
14     private boolean debug = false;                // Print debugging info?
15
16     public final Dimension minimumSize()
17     {
18         String test = "Screen 00 of ~00";
19         return new Dimension
20             (   controlSize + 4                    // marker width
21             +   (   panel != null
22                 ?   panel.textMetrics.stringWidth(test)
23                 :   panel.readerEnWidth * test.length()
24                 )
25             ,   Math.max
26                 (   panel.footerTextHeight + 1
27                 ,   controlSize + 4                // marker height
28                 )
29             );
30     }
31
32     public boolean mouseUp(Event evt, int x, int y)
33     {
34         // dbg("_p_pm_db.mouseUp()");
35
36         if (aheadRect.inside(x, y))
37         {
38             frame.doViewerRight();
39             return true;
40         }
41         else
42         if (closeRect.inside(x, y))
43         {
44             frame.ez_HTML.disposeFrame(frame);
45             return true;
46         }
47         else
48         if (abackRect.inside(x, y))
49         {
50             frame.doViewerLeft();
51             return true;
52         }
53         return false;
54     }
55
56     public final void paint(Graphics g)
57     {
58         Rectangle r = this.bounds();
59
60         if (g != null)
61         {
62             g.setColor(panel.getBackground());
63             g.fillRect(0, 0, r.width, r.height);
64             g.draw3DRect(0, 0, r.width - 1, r.height - 1, false);
65             g.setColor(panel.getForeground());
66             g.setFont(panel.readerFooterFont);
67         }
68
69         controlSize = panel.footerTextHeight*2/3;
70         setMarkers(g);

```

```
1      paintMarkers(g);
2
3      StringBuffer pI = new StringBuffer();
4      pI.append("Screen ").append(String.valueOf(panel.readerPageNum)).append(" of
5  ");
6      if (panel.readerNumPages == 0) pI.append("-");
7      pI.append(String.valueOf(panel.readerEstPages));
8      String pageInfo = pI.toString();
9
10     int infoWidth = getFontMetrics(panel.readerFooterFont).stringWidth(pageInfo);
11
12     int xLoc = (r.width - infoWidth)/2;
13     int yLoc = r.height - panel.footerTextHeight/5 - 1;
14     if (g != null) g.drawString(pageInfo, xLoc, yLoc);
15 }
16
17 public final void paintMarkers(Graphics g)
18 {
19     if (g == null) return;
20
21     Rectangle r = this.bounds();
22
23     g.setColor(panel.getForeground());
24
25     if (panel.readerPageNum > 1) // left arrow
26     {
27         int x = abackRect.x + 2
28         ,   y = abackRect.y + abackRect.height/2;
29         for (int arrow = 0; arrow < controlSize; arrow++)
30         {
31             g.drawLine(x + arrow, y - arrow/2, x + arrow, y + arrow/2);
32         }
33     }
34
35     if (panel.readerFrame) // close box
36     {
37         int x = closeRect.x + 2
38         ,   y = closeRect.y + 2
39         ,   w = closeRect.width - 5
40         ,   h = closeRect.height - 5;
41
42         g.drawRect( x, y, w, h);
43         g.drawLine( x, y, x+w, y+h);
44         g.drawLine(x+w, y, x, y+h);
45     }
46
47     if (panel.readerPageNum < panel.readerEstPages) // right arrow
48     {
49         int x = aheadRect.x + aheadRect.width - 2
50         ,   y = abackRect.y + abackRect.height/2;
51         for (int arrow = 0; arrow < controlSize; arrow++)
52         {
53             g.drawLine(x - arrow, y + arrow/2, x - arrow, y - arrow/2);
54         }
55     }
56 }
57
58 public final Dimension preferredSize()
59 {
60     return minimumSize();
61 }
62
63 public void setMarkers(Graphics g)
64 {
65     Rectangle r = this.bounds();
66
67     int w = this.size().width;
68     int h = this.size().height;
69
70     abackRect.reshape
```



```

1      ( w - 2 - (controlSize+4)*3
2      , h - 4 - controlSize
3      , controlSize + 4
4      , controlSize + 4
5      );
6
7      closeRect.reshape
8      ( w - 2 - (controlSize+4)*2
9      , h - 4 - controlSize
10     , controlSize + 4
11     , controlSize + 4
12     );
13
14     aheadRect.reshape
15     ( w - 2 - (controlSize+4)*1
16     , h - 4 - controlSize
17     , controlSize + 4
18     , controlSize + 4
19     );
20 }
21
22 public void update(Graphics g)
23 {
24     // g.setColor(getBackground());
25     // g.fillRect(0, 0, width, height);
26     // g.setColor(getForeground());
27
28     paint(g);
29 }
30
31 _p_pm_db(_p_rs_db panel, _f_rs_db frame)
32 {
33     this.panel = panel;
34     this.frame = frame;
35 }
36 }
37
38 /**
39  * %W% %E% Everett Stoub
40  *
41  * Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
42  *
43  * an object to store form element data
44  *
45  * this class is not downloaded unless needed
46  *
47  * @author Everett Stoub %I%, %G%
48  */
49 import java.awt.*;
50 import java.net.*;
51 import java.util.*;
52
53 public class _o_fe_db extends Object
54 {
55     public final static int NONE          = -1;
56     public final static int BUTTON        = 0;
57     public final static int CHECKBOX      = 1;
58     public final static int FILE          = 2;
59     public final static int HIDDEN        = 3;
60     public final static int IMAGE         = 4;
61     public final static int PASSWORD      = 5;
62     public final static int RADIO         = 6;
63     public final static int RESET         = 7;
64     public final static int SUBMIT        = 8;
65     public final static int TEXT          = 9;
66     public final static int SELECT        = 10;
67     public final static int TEXTAREA      = 11;
68
69     // state variables
70

```

```

1      public int    formNumber; // ez_HTML formsDB entry number (1,2,...)
2      public int    elemNumber; // the form's element number (1,2,...)
3      public int    tIndex;     // input element type index
4      public String  name;      // i/o:  element name to be passed to the
5      designated form-processing application
6
7      // twin value state variables: [0] - initial (for reset function) and [1] - current
8
9      public String  value[]     = new String[2]; // i/o:  value string(s) to be
10     passed to the designated form-processing application
11
12     public String  accept[]     = new String[2]; // state:  list of file types for
13     file input selection
14     public boolean checked[][] = new boolean[2][]; // state:  selector option or
15     checkbox or radio selection state: true for passing name & value(s)
16     public int    maxlength[] = new int[2]; // state:  number of accepted
17     characters to be passed from file, password, and text elements
18     public boolean multiple[] = new boolean[2]; // state:  multiple item
19     selection capability in selector options list
20
21     public URL     src[]        = new URL[2]; // input:  source url for image
22
23     public int     align[]      = new int[2]; // view:  index for
24     _p_rs_db.top, _p_rs_db.middle, or _p_rs_db.bottom image alignment
25     public int     border[]     = new int[2]; // view:  width of image border
26     inset in pixels
27     public int     cols[]       = new int[2]; // view:  number of displayed
28     characters across selector panes or textarea fields
29     public int     rows[]       = new int[2]; // view:  number of displayed
30     rows of textarea
31     public int     size[]       = new int[2]; // view:  number of displayed
32     characters in file, password, and test fields, or rows of selector panes
33
34     public _o_fe_db(int formNumber, int elemNumber, String tagText)
35     {
36         dbg("new _o_fe_db("+formNumber+", "+elemNumber+", [" + tagText + "])");
37
38         this.formNumber = formNumber;
39         this.elemNumber = elemNumber;
40         this.tIndex = _o_fe_db.getTagAttribTypeInt(tagText);
41         this.name = _o_tg_db.getTagAttribString(tagText, "name");
42
43         this.value[0] = this.value[1] = _o_tg_db.getTagAttribString(tagText,
44         "value");
45
46         this.accept[0] = this.accept[1] = _o_tg_db.getTagAttribString(tagText,
47         "accept");
48     }
49
50     // static methods
51
52     private static final void dbg(String s)
53     {
54         if (debug) System.out.println(s);
55     }
56     private static boolean debug = true; // Print debugging info?
57
58     public static final int getTagAttribTypeInt(String itsText)
59     {
60         String itsLabel = _o_tg_db.getTagAttribString(itsText, "type");
61
62         if (itsLabel == null)
63         {
64             return NONE;
65         }
66
67         String lowLabel = itsLabel.toLowerCase();
68
69         if (lowLabel.equals("button")) return BUTTON; else
70         if (lowLabel.equals("checkbox")) return CHECKBOX; else

```

```

1      if (lowLabel.equals("file"))      return FILE;      else
2      if (lowLabel.equals("hidden"))    return HIDDEN;    else
3      if (lowLabel.equals("image"))     return IMAGE;     else
4      if (lowLabel.equals("password"))  return PASSWORD; else
5      if (lowLabel.equals("radio"))     return RADIO;     else
6      if (lowLabel.equals("reset"))     return RESET;     else
7      if (lowLabel.equals("submit"))    return SUBMIT;    else
8      if (lowLabel.equals("text"))      return TEXT;      else
9      if (lowLabel.equals("select"))    return SELECT;    else
10     if (lowLabel.equals("textarea"))  return TEXTAREA; else
11
12     return NONE;
13 }
14 }
15
16 /**
17  *  %W% %E% Everett Stoub
18  *
19  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
20  *
21  *  an object to present html forms in panels
22  *
23  *  this class is not downloaded unless needed
24  *
25  *  @author Everett Stoub %I%, %G%
26  */
27 import java.awt.*;
28 import java.io.*;
29 import java.net.*;
30 import java.util.*;
31
32 public class _o_fm_db extends Object
33 {
34     public final static int NONE      = -1;
35     public final static int GET       = 0;
36     public final static int POST      = 1;
37
38     public final static int STIND     = 2;
39     public final static int TEXT      = 3;
40     public final static int DATA     = 4;
41
42     // state variables
43
44     public int mIndex;                // method index value
45     public int eIndex;                // encode index value
46     public String action;              // target for submit
47     command
48
49     public int formNumber;              // ez_HTML formsDB entry
50     number (1;2,...)
51     public Vector formElements;        // the elements of this
52     form
53
54     public String replaceContent(String formContent)
55     {
56         // create private form attribute: the form number
57
58         StringBuffer formsTag = new StringBuffer();
59         formsTag.append(_p_rs_db.PF)
60             .append(_p_rs_db.FM);        // private form tag
61
62         StringBuffer formAttr = new StringBuffer();
63         formAttr.append(_p_rs_db.BL)
64             .append(_p_rs_db.FN)        // private form attribute
65             .append(_p_rs_db.BQ)
66             .append(formNumber);        // form number insertion
67     string
68
69     String newContent = formContent;
70     String LcContent = newContent.toLowerCase();

```

```

1
2      // create private form element attributes: the form & element sequence numbers
3
4      int elemNumber = 0;
5      String insert[] =
6      (   _p_rs_db.IN                                // input elements
7      (buttons, etc.)
8      // ,   _p_rs_db.OP                                // selector option
9      elements
10     ,   _p_rs_db.ST                                // selector elements
11     ,   _p_rs_db.TA                                // textarea elements
12     );
13     for (int i = 0; i < insert.length; i++)
14     {
15         String insertion = _p_rs_db.PF + insert[i];
16         int j = LChContent.indexOf(insertion);           // the first of this kind
17     of tag
18         int k = LChContent.indexOf(_p_rs_db.SF, j);     // its terminus
19         while (-1 < j && j < k)
20         {
21             elemNumber++;
22
23             j += insertion.length();
24             k += _p_rs_db.SF.length();                 // post-tag content range
25     start point
26
27             // add this new element to this form
28
29             formElements.addElement
30             (   new _o_fe_db
31             (   formNumber
32             ,   elemNumber
33             ,   LChContent.substring(j,k)              // the entire element
34     input tag
35             )
36             );
37
38             StringBuffer elemAttr = new StringBuffer();
39             elemAttr.append(_p_rs_db.BL)
40             .append(_p_rs_db.EN)
41             .append(_p_rs_db.EQ)
42             .append(elemNumber)                        // element number
43     insertion string
44             .append(_p_rs_db.SF);
45
46             StringBuffer text = new StringBuffer();
47             text.append(newContent.substring(0,j))      // preceeding text,
48     including orig input tag preface
49             .append(formAttr.toString())              // private form attribute
50             .append(elemAttr.toString())              // private element
51     attribute
52             .append(newContent.substring(k));          // succeeding text,
53     excluding orig input tag suffix
54
55             newContent = text.toString();
56             LChContent = newContent.toLowerCase();
57
58             j = LChContent.indexOf(insertion, j);
59             k = LChContent.indexOf(_p_rs_db.SF, j);
60         }
61     }
62
63     StringBuffer finalTag = new StringBuffer();
64     finalTag.append(_p_rs_db.PF)
65     .append(_p_rs_db.SL)
66     .append(_p_rs_db.FM)                             // private end form tag
67     .append(_p_rs_db.SF);
68
69     StringBuffer output = new StringBuffer();
70     output.append(formTag.toString())

```

```
1      .append(formAttr.toString())
2      .append(_p_rs_db.SF)
3      .append(newContent)
4      .append(finalTag.toString());
5
6      return output.toString();
7  }
8
9  public int size()
10 {
11     return formElements.size();
12 }
13
14 public _o_fm_db(int formNumber, String formTag)
15 {
16     dbg("new _o_fm_db("+formNumber+",["+formTag+"]");
17
18     this.mIndex = _o_fm_db.getTagAttribMethodInt(formTag);
19     this.eIndex = _o_fm_db.getTagAttribEncodeInt(formTag);
20     this.action = _o_tg_db.getTagAttribString(formTag, "action");
21
22     this.formNumber = formNumber;
23     formElements = new Vector(10);
24 }
25
26 // static methods
27
28 private static final void dbg(String s)
29 {
30     if (debug) System.out.println(s);
31 }
32 private static boolean debug = true;    // Print debugging info?
33
34 public static int addForm
35 (   String tagText
36 ,   String itsText
37 ,   _p_rs_db panel
38 ,   int addIndexValue
39 ,   Graphics g
40 )
41 {
42     dbg("_o_fm_db.addForm(["+itsText+"],panel,"+addIndexValue+",g)");
43
44     int addIndex = addIndexValue;
45
46     int n = _o_tg_db.getTagAttribInt(tagText, _p_rs_db.FN, 0);
47     _o_fm_db form = panel.ez_HTML.getFormDB().getForm(n);
48     if (form == null)
49     {
50         return -addIndex;
51     }
52     else
53     {
54         return addIndex;
55     }
56 }
57
58 public static int addInput
59 (   String tagText
60 ,   String itsText
61 ,   _p_rs_db panel
62 ,   int addIndexValue
63 ,   Graphics g
64 )
65 {
66     dbg("_o_fm_db.addInput(["+itsText+"],panel,"+addIndexValue+",g)");
67
68     int addIndex = addIndexValue;
69
70     return addIndex;
```

```
1      )
2
3      public static int addOption
4      (   String tagText
5          ,   String itsText
6          ,   _p_rs_db panel
7          ,   int addIndexValue
8          ,   Graphics g
9      )
10     {
11         dbg("_o_fm_db.addOption(["+itsText+"] ,panel, "+addIndexValue+",g)");
12
13         int addIndex = addIndexValue;
14
15         return addIndex;
16     }
17
18     public static int addSelector
19     (   String tagText
20         ,   String itsText
21         ,   _p_rs_db panel
22         ,   int addIndexValue
23         ,   Graphics g
24     )
25     {
26         dbg("_o_fm_db.addSelector(["+itsText+"] ,panel, "+addIndexValue+",g)");
27
28         int addIndex = addIndexValue;
29
30         return addIndex;
31     }
32
33     public static int addTextArea
34     (   String tagText
35         ,   String itsText
36         ,   _p_rs_db panel
37         ,   int addIndexValue
38         ,   Graphics g
39     )
40     {
41         dbg("_o_fm_db.addTextArea(["+itsText+"] ,panel, "+addIndexValue+",g)");
42
43         int addIndex = addIndexValue;
44
45         return addIndex;
46     }
47
48     public static int endForm
49     (   String tagText
50         ,   String itsText
51         ,   _p_rs_db panel
52         ,   int addIndexValue
53         ,   Graphics g
54     )
55     {
56         dbg("_o_fm_db.endForm(["+itsText+"] ,panel, "+addIndexValue+",g)");
57
58         int addIndex = addIndexValue;
59
60         return addIndex;
61     }
62
63     public static int endOption
64     (   String tagText
65         ,   String itsText
66         ,   _p_rs_db panel
67         ,   int addIndexValue
68         ,   Graphics g
69     )
70     {
```

```
1      dbg("_o_fm_db.endOption(["+itsText+"],panel,"+addIndexValue+",g");
2
3      int addIndex = addIndexValue;
4
5      return addIndex;
6  }
7
8  public static int endSelector
9  (   String tagText
10     ,   String itsText
11     ,   _p_rs_db panel
12     ,   int addIndexValue
13     ,   Graphics g
14 )
15 {
16     dbg("_o_fm_db.endSelector(["+itsText+"],panel,"+addIndexValue+",g");
17
18     int addIndex = addIndexValue;
19
20     return addIndex;
21 }
22
23 public static int endTextArea
24 (   String tagText
25     ,   String itsText
26     ,   _p_rs_db panel
27     ,   int addIndexValue
28     ,   Graphics g
29 )
30 {
31     dbg("_o_fm_db.endTextArea(["+itsText+"],panel,"+addIndexValue+",g");
32
33     int addIndex = addIndexValue;
34
35     return addIndex;
36 }
37
38 public static final int getTagAttribEncodeInt(String itsText)
39 {
40     String itsLabel = _o_tg_db.getTagAttribString(itsText, "enctype");
41
42     if (itsLabel == null)
43     {
44         return NONE;
45     }
46
47     String lowLabel = itsLabel.toLowerCase();
48
49     if (lowLabel.equals("application/x-www-form-urlencoded"))    return STIND;
50 else
51     if (lowLabel.equals("text/plain"))                            return TEXT;
52 else
53     if (lowLabel.equals("multipart/form-data"))                  return DATA;
54 else
55
56     return NONE;
57 }
58
59 public static final int getTagAttribMethodInt(String itsText)
60 {
61     String itsLabel = _o_tg_db.getTagAttribString(itsText, "method");
62
63     if (itsLabel == null)
64     {
65         return NONE;
66     }
67
68     String lowLabel = itsLabel.toLowerCase();
69
70     if (lowLabel.equals("get"))    return GET;    else
```

```
1         if (lowLabel.equals("post"))    return POST;    else
2
3         return NONE;
4     }
5 }
6
7 /**
8  *  %W% %E% Everett Stoub
9  *
10  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
11  *
12  *  an object to maintain html form objects list
13  *
14  *  this class is not downloaded unless needed
15  *
16  *  @author Everett Stoub %I%, %G%
17  */
18 import java.awt.*;
19 import java.util.*;
20
21 public class _o_fm_db extends Object
22 {
23     public Vector formObjects;
24
25     private ez_html ez_HTML;
26
27     public _o_fm_db getForm(int formNumber)
28     {
29         if (0 < formNumber && formNumber <= formObjects.size())
30             return (_o_fm_db) formObjects.elementAt(formNumber-1);
31         else
32             return null;
33     }
34
35     public void newForm(_o_fm_db newForm)
36     {
37         formObjects.addElement(newForm);
38     }
39
40     public _o_fm_db(ez_html ez_HTML)
41     {
42         formObjects = new Vector(10);
43
44         this.ez_HTML = ez_HTML;
45     }
46 }
47
48
49 /**
50  *  %W% %E% Everett Stoub
51  *
52  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
53  *
54  *  a frame for displaying an image, with static methods for handling images
55  *
56  *  this class is not downloaded unless needed
57  *
58  *  @author Everett Stoub %I%, %G%
59  */
60 import java.awt.*;
61 import java.io.*;
62 import java.net.*;
63
64 public class _f_im_db extends Frame
65 {
66     Image itsImage = null;
67     private ez_html ez_HTML = null;
68     public final static String DI = "Release to display image...";
69     public final static String DM = "Drag to move image...";
70 }
```



```
1    int xImg = 0, xDown = 0, xWide = 0;
2    int yImg = 0, yDown = 0, yTall = 0;
3
4    private static final void dbg(String s)
5    {
6        if (debug) System.out.println(s);
7    }
8    private static boolean debug = false;        // Print debugging info?
9
10   public final boolean handleEvent(Event evt)
11   {
12       switch (evt.id)
13       {
14           case Event.WINDOW_DEICONIFY:    this.show();    break;
15           case Event.WINDOW_ICONIFY:      this.hide();    break;
16           case Event.WINDOW_DESTROY:      itsImage.flush();
17                                           this.dispose(); break;
18       }
19       return super.handleEvent(evt);
20   }
21
22   public final Dimension minimumSize()
23   {
24       Dimension itsSize = getToolkit().getScreenSize();
25       itsSize.width = Math.min(itsSize.width, xWide);
26       itsSize.height = Math.min(itsSize.height - 50, yTall); // hack for menus
27
28       return itsSize;
29   }
30
31   public boolean mouseDown(Event evt, int x, int y)
32   {
33       // dbg("_f_in_db.mouseDown()");
34
35       if (itsImage != null)
36           ez_HTML.showStatus(DM);
37
38       xDown = x;
39       yDown = y;
40
41       return true;
42   }
43
44   public boolean mouseDrag(Event evt, int x, int y)
45   {
46       if (itsImage != null)
47           ez_HTML.showStatus(DI);
48
49       xImg = Math.min(0, xImg + x - xDown);
50       yImg = Math.min(0, yImg + y - yDown);
51
52       xDown = x;
53       yDown = y;
54
55       repaint();
56
57       return true;
58   }
59
60   public boolean mouseEnter(Event evt, int x, int y)
61   {
62       ez_HTML.showStatus(DM);
63
64       return true;
65   }
66
67   public boolean mouseExit(Event evt, int x, int y)
68   {
69       ez_HTML.showStatus(_p_rs_db.MT);
```

```

1      return false;
2    }
3
4    public boolean mouseMove(Event evt, int x, int y)
5    {
6        ez_HTML.showStatus(DM);
7
8        return true;
9    }
10
11    public final void paint(Graphics g)
12    {
13        if (itsImage != null && g != null)
14            g.drawImage(itsImage, xImg, yImg, this);
15    }
16
17    public final Dimension preferredSize()
18    {
19        return minimumSize();
20    }
21
22    public _f_in_db(String itsTitle, Image itsImage, ez_html ez_HTML)
23    {
24        this.itsImage = itsImage;
25        this.ez_HTML = ez_HTML;
26
27        xWide = itsImage.getWidth(this);
28        yTall = itsImage.getHeight(this);
29
30        this.setTitle(itsTitle);
31        this.pack();
32        this.show();
33    }
34 }
35
36 /**
37  *  %W% %E% Everett Stoub
38  *
39  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
40  *
41  *  an object to load images from files
42  *
43  *  this class is not downloaded unless needed
44  *
45  *  @author Everett Stoub %I%, %G%
46  */
47 import java.awt.*;
48 import java.net.*;
49 import java.io.*;
50 import java.util.*;
51
52 public class _o_in_db extends Object
53 {
54     private ez_html      ez_HTML      = null;
55
56     private Vector      imageFileNames = null;      // vector of all image
57     file names for document
58     private Vector      graphicsURLs   = null;      // vector of all graphic
59     image URLs for document
60     private Vector      graphicsImages = null;      // vector of all graphic
61     images for document
62
63     public static int placeImage
64     (   String tagText
65     ,   ez_html ez_HTML
66     ,   _p_rs_db panel
67     ,   int addIndexValue
68     ,   Graphics g
69     )
70     {

```

```

1      String theFileName = _o_tg_db.getTagAttribString(tagText, panel.SR);
2      int addIndex = addIndexValue;
3
4      if (theFileName != null)
5      {
6          Image theImage = ez_HTML.getImageDB().fetchImage(theFileName);
7
8          int theWidth = 0;
9          if (theImage != null)
10         {
11             int theAlignment = _o_tg_db.getTagAlignment(tagText, panel.BOTTOM);
12             theWidth = addImage(theImage, theFileName, theAlignment,
13 panel.tagLength, ez_HTML, panel);
14             panel.tagLength = 0; // value saved in buffer
15         vectors
16         }
17         if (panel.te + theWidth
18             >= panel.xe - panel.indent(true)) // it won't fit on this line:
19         draw pending line
20         {
21             addIndex += panel.paintRow(g,1); // try to punch out pending
22         line elements
23             while (panel.yb > panel.ye) // start a new column?
24             {
25                 if (panel.lastColumn)
26                 {
27                     return -addIndex; // no more columns left to
28                 fill
29                 }
30                 panel.nextColumn();
31             }
32             panel.te = panel.xb + panel.indent(false) + theWidth;
33         }
34         else
35         {
36             panel.te += theWidth;
37         }
38         if (panel.te
39             == panel.xe - panel.indent(true)) // it fills this line: draw
40         it out
41         {
42             addIndex += panel.paintRow(g,0); // try to punch out pending
43         line elements
44             while (panel.yb > panel.ye) // start a new column?
45             {
46                 if (panel.lastColumn)
47                 {
48                     return -addIndex; // no more columns left to
49                 fill
50                 }
51                 panel.nextColumn();
52             }
53             panel.te = panel.xb + panel.indent(false);
54         }
55     }
56     return addIndex;
57 }
58
59 private static final int addImage
60 {
61     Image theImage
62     , String theFileName
63     , int theAlignment
64     , int tagLength
65     , ez_html ez_HTML
66     , _p_rs_db panel
67 }
68 {
69     int wIm = theImage.getWidth(ez_HTML);
70     int hIm = theImage.getHeight(ez_HTML);

```

```

1      int wC1 = panel.effectiveWidth();           // available width of the
2      column
3
4      if (wIm == 0 || hIm == 0) return 0;         // nothing to draw
5
6      if (wIm > wC1)                               // scale graphic down to
7      column width
8      {
9          hIm = Math.round( (float)wC1 * (float)hIm / (float)wIm );
10         wIm = wC1;
11     }
12
13     if (hIm > panel.readerRect.height)           // scale graphic down to
14     column height
15     {
16         wIm = Math.round( (float)panel.readerRect.height * (float)wIm / (float)hIm
17     );
18         hIm = panel.readerRect.height;
19     }
20
21     Image anOffImage = ez_HTML.createImage(wIm,hIm);
22     Graphics itsGraphic = anOffImage.getGraphics();
23
24     // itsGraphic.setColor(panel.getBackground());
25     // itsGraphic.fillRect(0, 0, wIm, hIm);
26     itsGraphic.drawImage(theImage, 0, 0, wIm, hIm, panel);
27     itsGraphic.dispose();
28
29     int maxAscent = hIm;                           // default case (BOTTOM)
30     if (theAlignment == panel.MIDDLE)
31         maxAscent = hIm/2; else
32     if (theAlignment == panel.TOP)
33         maxAscent = panel.textHalfSpace*3/2;       // cases LEFT & RIGHT are
34     not handled yet
35
36     panel.lineImages.addElement(anOffImage);
37     panel.dataValues.addElement
38     ( new Rectangle
39     ( tagLength                               // "x"      increment
40     for endIndex (character count)
41     , maxAscent                               // "y"      for
42     vertical placement of image (ascent) above baseline
43     , wIm                                     // "width"   element
44     width on line
45     , hIm - maxAscent                         // "height"  element
46     drop below baseline (descent)
47     )
48     );
49     panel.wordBlocks.addElement(theFileName);       // image cross reference
50     panel.hyperLinks.addElement(panel.hyperHead);
51     panel.styleCodes.addElement(panel.styleCode()); // ignored in rendering
52     images
53     panel.wordColors.addElement(panel.textColor);
54     panel.charsFonts.addElement(panel.textFont);
55
56     return wIm;                                   // the width of this
57     element on the line
58 }
59
60 public Image fetchImage(String fileName)
61 {
62     if (fileName == null)
63     {
64         return null;
65     }
66     else
67     {
68         Image theImage = null;
69

```

```

1      int imageDatabaseIndex = // get reference to image
2      database file name
3          imageFileNames.indexOf(fileName);
4
5      if (imageDatabaseIndex < 0) // not registered yet:
6      add it in
7      {
8          URL theURL = null;
9
10         try // look in the
11         documentBase
12         {
13             theURL = new URL(ez_HTML.getDocumentBase(), fileName);
14             InputStream is = theURL.openStream(); // test connection
15             theImage = ez_HTML.getImage(theURL);
16             ez_HTML.getTracker().addImage(theImage,0);
17
18             // jump-start image loading
19
20             boolean status = ez_HTML.getTracker().checkID(0,true);
21             try
22             {
23                 ez_HTML.getTracker().waitForID(0);
24             }
25             catch ( InterruptedException ie ) { }
26             Image bufferedImage = ez_HTML.createImage(1,1);
27             if (bufferedImage != null)
28             {
29                 Graphics bg = bufferedImage.getGraphics();
30                 if (bg != null)
31                 {
32                     bg.drawImage(theImage,0,0,this.ez_HTML);
33                     bg.dispose();
34                 }
35             }
36             bufferedImage.flush();
37         }
38         catch (MalformedURLException u)
39         {
40             System.out.println("_o_im_db.fetchImage("+fileName+"):
41             MalformedURLException = "+u);
42             theImage = null;
43         }
44         catch (IOException io)
45         {
46             System.out.println("_o_im_db.fetchImage("+fileName+"): InputStream
47             IOException = "+io);
48             theImage = null;
49         }
50
51         // update image database for document
52
53         imageFileNames.addElement(fileName); // the filename is the
54         main key to image database access
55         graphicsURLs.addElement(theURL); // in case we need to
56         download it again later
57         graphicsImages.addElement(theImage); // this may be reset to
58         null with imageUpdate
59     }
60     else
61     {
62         theImage = (Image)graphicsImages.elementAt(imageDatabaseIndex);
63     }
64     return theImage;
65 }
66
67 }
68
69 public static final void zoomImage // new frame with full-
70 size rendering

```

```

1      ( String theFileName
2      ,   ez_html ez_HTML
3      )
4      {
5          Image theImage = ez_HTML.getImageDB().fetchImage(theFileName);
6
7          if (theImage != null)
8          {
9              _f_im_db itsFrame = new _f_im_db(theFileName, theImage, ez_HTML);
10             itsFrame.setBackground(ez_HTML.getBackground());
11             itsFrame.setForeground(ez_HTML.getForeground());
12         }
13     }
14
15     public _o_im_db(ez_html ez_HTML)
16     {
17         this.ez_HTML = ez_HTML;
18
19         imageFileNames = new Vector(10);           // must be created before
20         calls to fetchImage
21         graphicsURLs    = new Vector(10);           // must be created before
22         calls to fetchImage
23         graphicsImages  = new Vector(10);           // must be created before
24         calls to fetchImage
25     }
26 }
27
28 /**
29  *  %W% %E% Everett Stoub
30  *
31  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
32  *
33  *  a frame for displaying a table
34  *
35  *  this class is not downloaded unless needed
36  *
37  *  @author Everett Stoub %I%, %G%
38  */
39 import java.awt.*;
40 import java.io.*;
41 import java.net.*;
42
43 public class _f_tb_db extends Frame
44 {
45     private boolean fullSize = true;                // appearance switch
46
47     public final static String DT = "Drag to move table...";
48     public final static String RT = "Release to view table...";
49     public final static String WT = "Wait for new table view...";
50
51     private _p_tb_db itsTable = null;
52     private ez_html ez_HTML = null;
53
54     private int xTbl = 0, xDown = 0;
55     private int yTbl = 0, yDown = 0;
56
57     public boolean justDragging = false;
58
59     private static final void dbg(String s)
60     {
61         if (debug) System.out.println(s);
62     }
63     private static boolean debug = false;           // Print debugging info?
64
65     public final boolean handleEvent(Event evt)
66     {
67         switch (evt.id)
68         {
69             case Event.WINDOW_DEICONIFY:            this.show();    break;
70             case Event.WINDOW_ICONIFY:               this.hide();    break;

```

```
1      case Event.WINDOW_DESTROY:      this.dispose(); break;
2      }
3      return super.handleEvent(evt);
4  }
5
6  public final Dimension minimumSize()
7  {
8      Dimension s = getToolkit().getScreenSize();
9      if (debug)
10     {
11         s.width /= 2;
12         s.height /= 2;
13     }
14     else
15     {
16         s.height -= 50; // hack for menus
17     }
18
19     Dimension t = itsTable.minimumSize();
20
21     s.width = Math.min(s.width, t.width);
22     s.height = Math.min(s.height, t.height);
23
24     // dbg("_f_tb_db.minimumSize(): screen
25     "+getToolkit().getScreenSize().width+"x"+getToolkit().getScreenSize().height+", min
26     table "+t.width+"x"+t.height+", min frame "+s.width+"x"+s.height);
27
28     return s;
29 }
30
31 public boolean mouseDown(Event evt, int x, int y)
32 {
33     // dbg("_f_tb_db.mouseDown("+x+", "+y+")");
34
35     if (itsTable != null)
36         ez_HTML.showStatus(DT);
37
38     xTbl = itsTable.location().x;
39     yTbl = itsTable.location().y;
40
41     xDown = x;
42     yDown = y;
43
44     justDragging = false;
45
46     return true;
47 }
48
49 public boolean mouseDrag(Event evt, int x, int y)
50 {
51     xTbl += x - xDown;
52     yTbl += y - yDown;
53
54     xDown = x;
55     yDown = y;
56
57     if (itsTable != null)
58     {
59         ez_HTML.showStatus(RT);
60
61         itsTable.move(xTbl, yTbl);
62     }
63     justDragging = true;
64     repaint();
65
66     return true;
67 }
68
69 public boolean mouseEnter(Event evt, int x, int y)
70 {
```

```

1      ez_HTML.showStatus(DT);
2
3      return true;
4  }
5
6  public boolean mouseExit(Event evt, int x, int y)
7  {
8      ez_HTML.showStatus(_p_rs_db.MT);
9
10     return false;
11 }
12
13 public boolean mouseMove(Event evt, int x, int y)
14 {
15     ez_HTML.showStatus(DT);
16
17     return true;
18 }
19
20 public boolean mouseUp(Event evt, int x, int y)
21 {
22     dbg("_f_tb_db.mouseUp()");
23
24     if (justDragging)
25     {
26         ez_HTML.showStatus(WT);
27         repaint();
28         justDragging = false;
29         return true;
30     }
31
32     return true;
33 }
34
35 public final Dimension preferredSize()
36 {
37     Dimension s = getToolkit().getScreenSize();
38     if (debug)
39     {
40         s.width /= 2;
41         s.height /= 2;
42     }
43     else
44     {
45         s.height -= 50; // hack for menus
46     }
47
48     Dimension t = itsTable.preferredSize();
49
50     s.width = Math.min(s.width, t.width * (fullSize?5:4)/4);
51     s.height = Math.min(s.height, t.height * (fullSize?5:4)/4);
52
53     // dbg("_f_tb_db.preferredSize(): screen
54     "+getToolkit().getScreenSize().width+"x"+getToolkit().getScreenSize().height+", pref
55     table "+t.width+"x"+t.height+", pref frame "+s.width+"x"+s.height);
56
57     return s;
58 }
59
60 public _f_tb_db(String itsTitle, _p_tb_db itsTable, _p_rs_db panel)
61 {
62     dbg("new _f_tb_db("+itsTitle+", "+itsTable.theTableNumber+")");
63
64     this.ez_HTML = panel.ez_HTML;
65     this.itsTable = itsTable;
66     itsTable.setFontSizeIndex(panel.readerSizeIndex);
67
68     this.setTitle(itsTitle);
69     this.setBackground(panel.getBackground());
70     this.setForeground(panel.getForeground());

```



```
1      this.setLayout(new BorderLayout());
2
3      Panel p = new Panel();           // help prevent WinOS header/footer
4  coverup
5      this.add("Center", p);
6
7      if (fullSize)                   // want overlarge tables to stretch
8  beyond window?
9      {
10         p.add(itsTable);             // default flow layout
11     }
12     else                             // or overlarge tables to shrink to
13  window area
14     {
15         p.setLayout(new BorderLayout());
16         p.add("Center", itsTable);
17     }
18
19     this.pack();
20     this.show();
21
22     xTbl = itsTable.location().x;
23     yTbl = itsTable.location().y;
24 }
25 }
26
27 /**
28  *  %W% %E% Everett Stoub
29  *
30  *  Copyright (c) '96-'97 ICON Web Products, Inc. All Rights Reserved.
31  *
32  *  an object to load tables from specifications
33  *
34  *  this class is not downloaded unless needed
35  *
36  *  @author Everett Stoub %I%, %G%
37  */
38  import java.awt.*;
39  import java.net.*;
40  import java.io.*;
41  import java.util.*;
42
43  public class _o_tb_db extends Object
44  {
45      private ez_html ez_HTML = null;
46      private Vector tableObjects = null;           // vector of all table objects
47  for document
48
49      public void addTable(_p_tb_db newTable)
50      {
51          tableObjects.addElement(newTable);
52      }
53
54      public _p_tb_db getCopy(int number, boolean readerFrame)
55      {
56          _p_tb_db orig = getTable(number);
57
58          if (orig == null)
59          {
60              return null;
61          }
62          else
63          {
64              _p_tb_db aCopy = orig.copy();
65              aCopy.readerFrame = readerFrame;
66              return aCopy;
67          }
68      }
69
70      private _p_tb_db getTable(int number)
```

```

1      {
2          if (number < 1 || number > tableObjects.size())
3          {
4              return null;
5          }
6          else
7          {
8              return (_p_tb_db)tableObjects.elementAt(number - 1);
9          }
10     }
11
12     public int size()
13     {
14         return tableObjects.size();
15     }
16
17     public _o_tb_db(ez_html ez_HTML)
18     {
19         this.ez_HTML = ez_HTML;
20         tableObjects = new Vector(10);
21     }
22
23     // static methods
24
25     private static final void dbg(String s)
26     {
27         if (debug) System.out.println(s);
28     }
29     private static boolean debug = false; // Print debugging info?
30
31     public static int placeTable
32     (   String tagText
33     ,   ez_html ez_HTML
34     ,   _p_rs_db panel
35     ,   int addIndexValue
36     ,   Graphics g
37     )
38     {
39         int addIndex = addIndexValue;
40         int tableNumber = _o_tg_db.getTagAttribInt(tagText, _p_rs_db.VL, 0);
41
42         // dbg("_o_tb_db.placeTable("+tableNumber+"");
43         panel(" "+panel.readerContent.substring(0,Math.min(30,panel.readerContent.length()))+(pa
44 nel.readerContent.length()<30?panel.ZZ.substring(panel.readerContent.length()):panel.M
45 T)+")");
46
47         if (tableNumber > 0)
48         {
49             _p_tb_db theTable = ez_HTML.getTableDB().getCopy(tableNumber,
50 panel.readerFrame);
51
52             if (theTable != null)
53             {
54                 addIndex += panel.paintRow(g,0); // try to punch out pending
55 line elements
56                 theTable.setBackground(panel.getBackground());
57                 theTable.setForeground(panel.getForeground());
58
59                 panel.add(theTable); // reshaping below
60                 panel.pageCorps.addElement(theTable); // for playback
61                 theTable.setFontSizeIndex(panel.readerSizeIndex - 1);
62
63                 Dimension c = panel.columnSize(); // as big as it can be
64                 Dimension p = theTable.preferredSize(); // as good as it can be
65                 Dimension m = theTable.minimumSize(); // as small as it can be
66                 Dimension t = new Dimension(c); // new table dimensions:
67 default to column
68

```

```

1      // dbg("_o_tb_db.placeTable("+tableNumber+"): panel col
2      "+c.width+"x"+c.height+", pref table "+p.width+"x"+p.height+", min table
3      "+m.width+"x"+m.height);
4
5      if (p.width < c.width)                // reduce width to preferred
6  width?
7          t.width = p.width;                // preferred width OK
8
9      if (p.height < c.height)              // reduce height to preferred
10 height?
11         t.height = p.height;              // preferred height OK
12
13         int clearH = panel.ye - panel.yb;  // headroom in column for
14 table
15         if (clearH < t.height)             // table height too tall?
16         {
17             if                            // room for nice first row?
18             ( theTable.prefRowHeight(0) < clearH
19             || m.height < clearH           // or room for all minimum
20 rows?
21             || 10*clearH > c.height        // or a full column won't
22 help (much)
23             )
24             {
25                 t.height = clearH;         // at least one row fits into
26 the headroom
27             }
28             else                           // this is the last column:
29 try harder
30                 if (!panel.lastColumn)    // just start a new column
31                 {
32                     panel.nextColumn();
33                 }
34             else                           // it'll have to wait for the
35 sequel
36                 {
37                     panel.remove(theTable); // drop it
38                     panel.pageComps.removeElementAt(panel.pageComps.size() - 1);
39                     return -addIndex;       // no more columns left to
40 fill
41                 }
42         }
43         // dbg("_o_tb_db.placeTable(): reshape table to "+t.width+"x"+t.height);
44         theTable.reshape
45         ( panel.xb + Math.max(0, (panel.effectiveWidth() - t.width)) / 2
46         , panel.yb
47         , t.width
48         , t.height
49         );
50         theTable.show();
51
52         panel.lineImages.addElement(null);
53         panel.dataValues.addElement
54         ( new Rectangle
55         ( panel.tagLength                // "x"      increment for
56 endIndex (character count)
57         , 0                             // "y"      for vertical
58 placement of image (ascent) above baseline
59         , t.width                        // "width"   element width
60 on line
61         , t.height                       // "height"  element drop
62 below baseline (descent)
63         )
64         );
65         panel.wordBlocks.addElement(theTable); // table reference
66         panel.hyperLinks.addElement(panel.hyperHead);
67         panel.styleCodes.addElement(panel.styleCode());
68         panel.wordColors.addElement(panel.textColor);
69         panel.charsFonts.addElement(panel.textFont);
70

```

```

1         panel.tagLength = 0;                                // value saved in buffer
2     vectors
3     }
4     }
5     return addIndex;
6 }
7
8     public static final void zoomTable                      // new frame with full-size
9 rendering
10     ( Integer theTableNumber
11     ,   _p_rs_db panel
12     )
13     {
14         // dbg("_o_tb_db.zoomTable("+theTableNumber+", "+ez_HTML+"");
15
16         int tableNumber = theTableNumber.intValue();
17         if (tableNumber <= panel.ez_HTML.getTableDB().size())
18         {
19             _p_tb_db theTable = panel.ez_HTML.getTableDB().getCopy(tableNumber, true);
20             String itsTitle =
21             (   theTable.theCaption == null
22             ?   "Table "+tableNumber
23             :   theTable.theCaption
24             );
25             _f_tb_db itsFrame = new _f_tb_db(itsTitle, theTable, panel);
26         }
27     }
28 }
29
30 /**
31  *  %W% %E% Everett Stoub
32  *
33  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
34  *
35  *  an object to store table row data
36  *
37  *  this class is not downloaded unless needed
38  *
39  *  @author Everett Stoub %I%, %G%
40  */
41 import java.awt.*;
42 import java.util.*;
43
44 public class _o_tr_db extends Object
45 {
46     public Color    rowBg    = null;
47     public int      align    = _p_rs_db.LEFT;
48     public int      vAlign   = _p_rs_db.CENTER;
49
50     public Vector    itsCells = new Vector();
51
52     public _o_tr_db
53     (   Color rowBg
54     ,   int align
55     ,   int vAlign
56     )
57     {
58         this.rowBg = rowBg;
59         this.align = align;
60         this.vAlign = vAlign;
61     }
62 }
63
64 /**
65  *  %W% %E% Everett Stoub
66  *
67  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
68  *
69  *  an object to handle html tables
70  *

```

```

1  * this class is not downloaded unless needed
2  *
3  * HTML tag parsing honors the following commands and attributes
4  *
5  *   • <table>...</table>          standard new table tag
6  *       align=type                left, center, right
7  *       bgcolor=color             rgb or color name
8  *       border=pixels            cell embossed border size
9  *       cellpadding=pixels       cell inset size
10 *       cellspacing=pixels       border embossement size
11 *       hspace=pixels            table width inset
12 *       valign=type              top, bottom
13 *       vspace=pixels            table height inset
14 *       width=pixels or %        table rel or abs width preference
15 *
16 *   • <tr>...</tr>                standard new row tag
17 *       align=type                left, center, right
18 *       bgcolor=color            rgb or color name
19 *       valign=type              top, bottom
20 *
21 *   • <td>...</td>                standard new column data tag
22 *   • <th>...</th>                standard new column header tag
23 *       align=type                left, center, right
24 *       bgcolor=color            rgb or color name
25 *       valign=type              top, center, bottom
26 *
27 *   • <caption>...</caption>      standard new row tag
28 *       align=type                top, bottom
29 *
30 * @author Everett Stoub %I%, %G%
31 */
32 import java.awt.*;
33 import java.awt.image.*;
34 import java.util.*;
35
36 public class _p_th_db extends Panel
37 {
38     public final static String EC      = "bgcolor";
39     public final static String EO      = "border";
40     public final static String CP      = "cellpadding";
41     public final static String CS      = "cellspacing";
42     public final static String HT      = "height";
43     public final static String SH      = "hspace";
44     public final static String SV      = "vspace";
45     public final static String TA      = "table";
46     public final static String TC      = "caption";
47     public final static String TD      = "td";
48     public final static String TH      = "th";
49     public final static String TR      = "tr";
50     public final static String VN      = "valign";
51     public final static String ZM      = "Click to zoom table...";
52     public final static String ZR      = "Release for fullsize table view...";
53     public final static String ZW      = "Wait for fullsize table view...";
54
55     public ez_html ez_HTML = null;
56     public Integer theTableNumber = new Integer(0);
57     public int fontSizeIndex = 2;
58     public boolean readerFrame = false;
59     public boolean tableSpecParsed = false;
60     public boolean layoutComplete = false;
61
62     private String tableSpec = null;
63
64     // table parameters
65
66     private int align = _p_rs_db.LEFT; // current setting
67     private int valign = _p_rs_db.CENTER;
68     private int hSpace = 0; // current setting
69     private int vSpace = 0; // current setting
70     private int border = 0; // current setting

```

```

1      private int      cellPadding = 0;           // current setting
2      private int      cellSpacing = 0;           // current setting
3      private int      minWidth = 0;              // current guideline for whole
4      table
5      private int      minHeight = 0;             // current guideline for whole
6      table
7
8      private Insets    insets = new Insets(0,0,0,0);
9      private Vector    tableRows = new Vector(); // each entry is a row object
10     private _p_tc_db   tableCells[][];           // array of _p_tc_db cell objects
11     private Dimension  tableSize = new Dimension();
12     private int        nRows = 1;
13     private int        nCols = 1;
14     private int        itsWidth = 0;
15     private int        itsHeight = 0;
16
17     public String      theCaption = null;
18     private int        capAlign = _p_rs_db.TOP;   // current setting
19
20     public String addCaption(boolean upperCaption)
21     {
22         if
23         ( ( theCaption == null)
24         || ( upperCaption && capAlign != _p_rs_db.TOP)
25         || (!upperCaption && capAlign == _p_rs_db.TOP)
26         )
27             return _p_rs_db.MT;
28
29         StringBuffer itsContent = new StringBuffer();
30
31         itsContent          // new break, with center formatting
32         .append(_p_rs_db.PF)
33         .append(_p_rs_db.BR).append(_p_rs_db.BL)
34         .append(_p_rs_db.AL).append(_p_rs_db.EQ)
35         .append(_p_rs_db.CJ)
36         .append(_p_rs_db.SF);
37         itsContent.append(theCaption);
38         itsContent          // end paragraph
39         .append(_p_rs_db.PF)
40         .append(_p_rs_db.SL)
41         .append(_p_rs_db.PA)
42         .append(_p_rs_db.SF);
43
44         // dbg("_p_tb_db("+theTableNumber+").addCaption("+upperCaption+"):
45         "+itsContent.toString());
46
47         return itsContent.toString();
48     }
49
50     private void addContent(String content)
51     {
52         // dbg("_p_tb_db("+theTableNumber+").addContent("+content+")");
53
54         if (theCaption == _p_rs_db.MT)           // load content as the caption
55         {
56             if (content == _p_rs_db.MT)           // preclude regression
57             {
58                 theCaption = null;                 // complete the load
59             }
60             else
61             {
62                 // dbg("... theCaption = "+content);
63                 theCaption = content;              // capture the load
64             }
65         }
66         else
67         if (tableRows.size() > 0)                 // content could be extraneous
68         {
69             _o_tr_db thisRow = (_o_tr_db)tableRows.lastElement();
70

```

```

1      if (thisRow.itsCells.size() > 0)
2      {   _p_tc_db thisCell = (_p_tc_db)thisRow.itsCells.lastElement();
3
4          //   dbg("... the cell(R"+tableRows.size()+"C"+thisRow.itsCells.size()+"") =
5      "+content");
6          thisCell.content += content;    //   capture the load
7      }
8      }
9  }
10
11  private void addElement(String itsText)
12  {
13      //   dbg("_p_tb_db("+theTableNumber+")addElement("+itsText+")");
14
15      String tagText = itsText.toLowerCase();
16
17      if (tagText.startsWith(TA))                                //   an
18      html new table tag w/ attributes
19      {
20          setBackground
21          (   _o_tg_db.parseColorValue
22              (   _o_tg_db.getTagAttribString(tagText, BC)
23                  ,   getBackground().getRGB()
24              )
25          );
26          align = _o_tg_db.getTagAlignment(tagText, align);
27          valign = _o_tg_db.getTagAlignment(tagText, VN, valign);
28
29          hSpace = _o_tg_db.getTagAttribInt(tagText, SH, hSpace);
30          vSpace = _o_tg_db.getTagAttribInt(tagText, SV, vSpace);
31
32          minWidth = _o_tg_db.getTagPerCentWidth(tagText);        //   coded
33      negative for per cent value, positive for pixel value
34          minHeight = _o_tg_db.getTagAttribInt(tagText, HT, minHeight);
35
36          if (tagText.indexOf(BO) > -1)
37              border = _o_tg_db.getTagAttribInt(tagText, BO, 1);
38          cellPadding = _o_tg_db.getTagAttribInt(tagText, CP, cellPadding);
39          cellSpacing = _o_tg_db.getTagAttribInt(tagText, CS, cellSpacing);
40      }
41      else if (tagText.startsWith(TR))                                //   an
42      html new row tag w/ attributes
43      {
44          //   dbg("... new table row: tagText = "+tagText);
45          _o_tr_db thisRow = new _o_tr_db
46          (   _o_tg_db.parseColorValue
47              (   _o_tg_db.getTagAttribString(tagText, BC)
48                  ,   getBackground().getRGB()
49              )
50              ,   _o_tg_db.getTagAlignment(tagText, align)
51              ,   _o_tg_db.getTagAlignment(tagText, VN, valign)
52          );
53          tableRows.addElement(thisRow);                                //   align,
54      bgcolor, valign, cells vector
55          nRows = tableRows.size();
56      }
57      else if (tagText.startsWith(TH))                                //   an
58      html column header tag w/ attributes for current row
59          || tagText.startsWith(TD))                                //   an
60      html column data tag w/ attributes for current row
61      {
62          _o_tr_db thisRow = (_o_tr_db)tableRows.lastElement();
63
64          _p_tc_db thisCell = new _p_tc_db                                //
65      background color
66          (   _o_tg_db.parseColorValue
67              (   _o_tg_db.getTagAttribString
68                  (   tagText, BC)
69                  ,   thisRow.rowBg.getRGB()
70              )

```

```

1      , getForeground()
2      , _o_tg_db.getTagAlignment(tagText, thisRow.align)           //
3      horizontal alignment
4      , _o_tg_db.getTagAlignment(tagText,VN,thisRow.vAlign)       //
5      vertical alignment
6      , _o_tg_db.getTagPerCentWidth(tagText)                     //
7      minimum cell width
8      , _o_tg_db.getTagAttribInt(tagText,"colspan",1)             // column
9      group value
10     , _o_tg_db.getTagAttribInt(tagText,"rowspan",1)              // row
11     group value
12     , _o_tg_db.getTagAttribInt(tagText,HT,23)                   //
13     minimum cell height
14     , border
15     , cellPadding
16     , cellSpacing
17     , tagText.startsWith(TD)                                     //
18     boolean isData
19     );
20     thisRow.itsCells.addElement(thisCell);
21     int rowCells = 0;
22     for (int i = 0; i < thisRow.itsCells.size(); i++)
23         rowCells += ((_p_tc_db)thisRow.itsCells.elementAt(i)).colSpan;
24     nCols = Math.max(nCols, rowCells);
25 }
26 else if (tagText.startsWith(_p_rs_db.TB))                        // a
27 private table in a cell tag
28 {
29     // dbg("... finding row "+tableRows.size()+" object");
30     _o_tr_db thisRow = (_o_tr_db)tableRows.lastElement();
31     // dbg("... finding cell "+thisRow.itsCells.size()+" object");
32     _p_tc_db thisCell = (_p_tc_db)thisRow.itsCells.lastElement();
33
34     thisCell.content += "<"+tagText+">";                          //
35     capture the load
36 }
37 else if (tagText.startsWith(TC))                                  // an
38 html caption tag w/ attributes
39 {
40     theCaption = _p_rs_db.MT;                                     // code:
41 next content value is caption
42     capAlign = _o_tg_db.getTagAlignment(tagText, capAlign);
43 }
44 else
45 if
46 ( tagText.startsWith(_p_rs_db.SL+TA)                            // an
47 html end new table tag w/ attributes
48 || tagText.startsWith(_p_rs_db.SL+TR)                            // an
49 html end new row tag w/ attributes
50 || tagText.startsWith(_p_rs_db.SL+TH)                            // an
51 html end column header tag w/ attributes for current row
52 || tagText.startsWith(_p_rs_db.SL+TD)                            // an
53 html end column data tag w/ attributes for current row
54 || tagText.startsWith(_p_rs_db.SL+_p_rs_db.TB)                   // a
55 private end table in a cell tag
56 || tagText.startsWith(_p_rs_db.SL+TC)                            // an
57 html end caption tag w/ attributes
58 )
59 {
60     // just ignore these
61 }
62 else
63 {
64     // dbg("... unhandled tagText = <"+tagText+">");
65
66     _o_tr_db thisRow = (_o_tr_db)tableRows.lastElement();
67     _p_tc_db thisCell = (_p_tc_db)thisRow.itsCells.lastElement();
68
69     thisCell.content += "<"+tagText+">";                          //
70     capture the load

```



```

1      )
2    }
3
4    public _p_tb_db copy()
5    {
6      // dbg("_p_tb_db"+"theTableNumber+").copy()");
7
8      _p_tb_db aCopy = new _p_tb_db
9      (   this.tableSpec
10      ,   this.theTableNumber
11      ,   this.ez_HTML
12      );
13
14      return aCopy;
15    }
16
17    private final void dbg(String s)
18    {
19      if (debug) System.out.println(s);
20    }
21    private boolean debug = false;           // Print debugging info?
22
23    public synchronized void doLayout()
24    {
25      // dbg("_p_tb_db"+"theTableNumber+").doLayout(): "+nRows+"x"+nCols+" table");
26
27      if (!tableSpecParsed)
28        parseTableSpec();
29
30      boolean zoomableCells = false;         // true if top level is _f_tb_db
31      Component ancestor = getParent();
32      while (ancestor != null)
33      {
34        if (ancestor instanceof _f_tb_db)
35          zoomableCells = true;
36        ancestor = ancestor.getParent();
37      }
38
39      tableCells = new _p_tc_db[nRows][nCols];
40      for (int j = 0; j < nRows; j++)        // assign row & column #'s to each
41      cell
42      {
43        _o_tr_db thisRow = (_o_tr_db)tableRows.elementAt(j);
44        int nCells = thisRow.itsCells.size();
45        for (int i = 0; i < nCells; i++)    // locate all cells in this row
46        {
47          _p_tc_db thisCell = (_p_tc_db)thisRow.itsCells.elementAt(i);
48          StringBuffer itsContent = new StringBuffer();
49          if (thisCell.content.length() > 0)
50          {
51            itsContent.append(_p_rs_db.PF).append(_p_rs_db.PA);
52            if (!thisCell.isData           // center formatting
53                || thisCell.align == _p_rs_db.CENTER)
54
55              itsContent.append(_p_rs_db.BL).append(_p_rs_db.AL).append(_p_rs_db.BQ).append(_p_r
56 s_db.CJ);
57
58              itsContent.append(_p_rs_db.SF);
59
60              if (!thisCell.isData)        // bold formatting
61
62                itsContent.append(_p_rs_db.PF).append(_p_rs_db.BD).append(_p_rs_db.SF).append(this
63 Cell.content).append(_p_rs_db.PF).append(_p_rs_db.SL).append(_p_rs_db.BD).append(_p_r
64 s_db.SF);
65
66                else                          // normal formatting
67                  itsContent.append(thisCell.content);
68
69              itsContent.append(_p_rs_db.PF).append(_p_rs_db.SL).append(_p_rs_db.PA).append(_p_r
70 s_db.SF);
71          }

```

```

1      _p_rs_db aPane = new _p_rs_db // a panel component
2      (      itsContent.toString() // specified html content
3      ,      ez_HTML // the applet
4      ,      null // no frame
5      ,      false // NOT the reader screen
6      ,      1 // maxColumns
7      ,      false // no paging indicator marks
8      ,      false // no page position footer
9      ,      thisCell.cellBg
10     ,      thisCell.cellFg
11     ,      (itsContent.length() > 0 ? border: 0)
12     ,      cellPadding
13     ,      cellSpacing
14     ,      zoomableCells // zoomability
15     );
16     aPane.setFontSizeIndex(this.fontSizeIndex);
17
18     int c = i, r = j;
19     while (tableCells[r][c] != null) // find first empty table cell (r,c)
20     if (++c == nCols) // start a new row!
21     {
22         c = 0; // start new row with 1st column
23         if (++r == nRows) // no more rows: attempt recovery
24         {
25             c = i; r = j;
26             break; // could replace prior cell
27         }
28     }
29     if (tableCells[r][c] == null) // prevent cell replacement
30     {
31         thisCell.colNum = c; thisCell.rowNum = r;
32         for (int f = 0; f < thisCell.rowSpan && r + f < nRows; f++)
33             for (int g = 0; g < thisCell.colSpan && c + g < nCols; g++)
34                 tableCells[r + f][c + g]
35                     = thisCell; // fill the element's footprint
36     }
37     this.add(thisCell); // add the cell to this table panel
38 (reshape later)
39     thisCell.add("Center", aPane); // add the reader panel to the cell
40     thisCell.show();
41 }
42
43 layoutComplete = true;
44 this.reshapeAll();
45 this.invalidate();
46 this.validate();
47 }
48
49 public Insets insets()
50 {
51     return insets;
52 }
53
54 private int minColLoc(int c)
55 {
56     int x = insets.left;
57     for (int i = 0; i < c; i++)
58         x += minColWidth(i);
59
60     return x;
61 }
62
63 private int minColLoc(_p_tc_db c)
64 {
65     return minColLoc(c.colNum);
66 }
67
68 private int minColWidth(int c)
69 {
70     int minColWidth = 0;

```

```

1      for (int r = 0; r < nRows; r++)
2          if (tableCells[r][c] != null)
3              minColWidth = Math.max
4                  ( minColWidth
5                    , tableCells[r][c].minimumSize().width
6                    / tableCells[r][c].colSpan
7                  );
8
9      return minColWidth;
10 }
11
12 private int minColWidth(_p_tc_db c)
13 {
14     return minColWidth(c.colNum);
15 }
16
17 public Dimension minimumSize()
18 {
19     if (!layoutComplete)
20         doLayout();
21
22     //
23     dbg("_p_tc_db"+"theTableNumber+").minimumSize()+"minTableSize().width+", "+minTableS
24     ize().height+"");
25
26     return minTableSize();
27 }
28
29 public int minRowHeight(int r)
30 {
31     int minRowHeight = 0;
32     for (int c = 0; c < nCols; c++)
33         if (tableCells[r][c] != null)
34             minRowHeight = Math.max
35                 ( minRowHeight
36                   , tableCells[r][c].minimumSize().height
37                   / tableCells[r][c].rowSpan
38                 );
39
40     return minRowHeight;
41 }
42
43 private int minRowHeight(_p_tc_db c)
44 {
45     return minRowHeight(c.rowNum);
46 }
47
48 private int minRowLoc(int r)
49 {
50     int y = insets.top;
51     for (int j = 0; j < r; j++)
52         y += minRowHeight(j);
53
54     return y;
55 }
56
57 private int minRowLoc(_p_tc_db c)
58 {
59     return minRowLoc(c.rowNum);
60 }
61
62 public Dimension minTableSize()
63 {
64     Dimension d = new Dimension();
65
66     for (int c = 0; c < nCols; c++)
67         d.width += minColWidth(c);
68
69     for (int r = 0; r < nRows; r++)
70         d.height += minRowHeight(r);

```

```

1
2     d.width += insets.left + insets.right;
3     d.height += insets.top + insets.bottom;
4
5     return d;
6 }
7
8 public boolean mouseDown(Event evt, int x, int y)
9 {
10    // dbg("_p_tb_db.mouseDown("+x+", "+y+"");
11
12    if (getParent() != null)
13    {
14        if (getParent() instanceof _p_rs_db)
15        {
16            ez_HTML.showStatus(ZR);
17            return true;
18        }
19        return false;
20    }
21
22    return true;
23 }
24
25 public boolean mouseEnter(Event evt, int x, int y)
26 {
27     if (getParent() instanceof _p_rs_db)
28         ez_HTML.showStatus(ZM);
29
30     return true;
31 }
32
33 public boolean mouseMove(Event evt, int x, int y)
34 {
35     if (getParent() instanceof _p_rs_db)
36         ez_HTML.showStatus(ZM);
37
38     return true;
39 }
40
41 public boolean mouseExit(Event evt, int x, int y)
42 {
43     ez_HTML.showStatus(_p_rs_db.MT);
44
45     return true;
46 }
47
48 public boolean mouseUp(Event evt, int x, int y)
49 {
50    // dbg("_p_tb_db.mouseUp()");
51
52    if (getParent() != null)
53    {
54        if (getParent() instanceof _p_rs_db)
55        {
56            // dbg("_p_tb_db.mouseUp(): zooming table "+this.theTableNumber);
57
58            ez_HTML.showStatus(ZW);
59            _o_tb_db.zoomTable
60            (    this.theTableNumber
61              ,    (_p_rs_db).getParent()
62              );
63            return true;
64        }
65        else
66        {
67            return false;
68        }
69    }
70

```

```

1:         return super.mouseUp(evt, x, y);
2:     }
3:
4:     public synchronized void paint(Graphics g)
5:     {
6:         // dbg("_p_tb_db"+"theTableNumber+", "+countComponents()+" components:
7:         "+size().width+" x "+size().height+").paint(g); isEnabled = "+isEnabled()+"", isShowing
8:         = "+isShowing()+"", isValid = "+isValid()+"", isVisible = "+isVisible()+"",
9:         layoutComplete = "+layoutComplete);
10:
11:         if (!layoutComplete)
12:             doLayout();
13:
14:         Rectangle r = bounds();
15:
16:         if (g != null)
17:         {
18:             g.setColor(getBackground());
19:             g.fillRect(0, 0, r.width, r.height);
20:             for (int i = 0; i < border; i++)
21:                 g.draw3DRect
22:                 (
23:                     i + hSpace + cellSpacing/2
24:                     , i + vSpace + cellSpacing/2
25:                     , r.width - 1 - 2*hSpace - cellSpacing - 2*i
26:                     , r.height - 1 - 2*vSpace - cellSpacing - 2*i
27:                     , true
28:                 );
29:             g.setColor(getForeground());
30:         }
31:
32:     private void parseTableSpec()
33:     {
34:         StringTokenizer tagBlocks = new StringTokenizer           // break up the text
35:         into normal tagBlocks
36:         (tableSpec, _p_rs_db.PF, true);                          // final true returns
37:         token on next call
38:
39:         while (tagBlocks.hasMoreTokens())                        // tag block to
40:         follow
41:         {
42:             String tagBlock = tagBlocks.nextToken(_p_rs_db.PF); // [<tag
43:             text>content], [<tag text>content], ..., [<tag text>]
44:
45:             // handle all html table instructions
46:
47:             if (tagBlock.equals(_p_rs_db.PF))                    // an html "<" (PF)
48:             tag prefix?
49:             {
50:                 String tagText =
51:                 tagBlocks.nextToken(_p_rs_db.SF);                // should be tag text
52:
53:                 if (tagBlocks.hasMoreTokens())
54:                 tagBlock = tagBlocks.nextToken();                // should be ">" (SF)
55:             tag suffix
56:             else
57:                 tagBlock = _p_rs_db.MT;                          // syntax error
58:
59:                 if (tagBlock.equals(_p_rs_db.SF))                // the normal case
60:                 {
61:                     addElement(tagText);
62:                     tagBlock = _p_rs_db.MT;                        // tagText handled,
63:                 transfer text capture
64:                 }
65:                 else
66:                 syntax error
67:                 {
68:                     tagBlock = _p_rs_db.PF + tagText + tagBlock; // restore tagBlock
69:                 to body text
70:                 }

```

```

1      )
2
3      // handle all html text content
4
5      if (!tagBlock.equals(_p_rs_db.PF)
6      && !tagBlock.equals(_p_rs_db.MT)) // not an html tag?
7      must be content
8      {
9          addContent(tagBlock);
10     }
11 }
12     tableSpecParsed = true;
13 }
14
15 private int prefColLoc(int c)
16 {
17     int x = insets.left;
18     for (int i = 0; i < c; i++)
19         x += prefColWidth(i);
20
21     return x;
22 }
23
24 private int prefColLoc(_p_tc_db c)
25 {
26     return prefColLoc(c.colNum);
27 }
28
29 private int prefColWidth(int c)
30 {
31     int prefColWidth = 0;
32     for (int r = 0; r < nRows; r++)
33         if (tableCells[r][c] != null)
34             prefColWidth = Math.max
35             ( prefColWidth
36              , tableCells[r][c].preferredSize().width
37              / tableCells[r][c].colSpan
38             );
39
40     return prefColWidth;
41 }
42
43 private int prefColWidth(_p_tc_db c)
44 {
45     return prefColWidth(c.colNum);
46 }
47
48 public Dimension preferredSize()
49 {
50     if (!layoutComplete)
51         doLayout();
52
53     //
54     dbg("_p_tb_db"+"theTableNumber+").preferredSize()+"prefTableSize().width+", "prefTa
55 bleSize().height+");
56
57     return prefTableSize();
58 }
59
60 public int prefRowHeight(int r)
61 {
62     int prefRowHeight = 0;
63     for (int c = 0; c < nCols; c++)
64         if (tableCells[r][c] != null)
65             prefRowHeight = Math.max
66             ( prefRowHeight
67              , tableCells[r][c].preferredSize().height
68              / tableCells[r][c].rowSpan
69             );
70

```

```
1      return prefRowHeight;
2  }
3
4  private int prefRowHeight(_p_tc_db c)
5  {
6      return prefRowHeight(c.rowNum);
7  }
8
9  private int prefRowLoc(int r)
10 {
11     int y = insets.top;
12     for (int j = 0; j < r; j++)
13         y += prefRowHeight(j);
14
15     return y;
16 }
17
18 private int prefRowLoc(_p_tc_db c)
19 {
20     return prefRowLoc(c.rowNum);
21 }
22
23 public Dimension prefTableSize()
24 {
25     Dimension d = new Dimension();
26
27     for (int c = 0; c < nCols; c++)
28         d.width += prefColWidth(c);
29
30     for (int r = 0; r < nRows; r++)
31         d.height += prefRowHeight(r);
32
33     d.width += insets.left + insets.right;
34     d.height += insets.top + insets.bottom;
35
36     return d;
37 }
38
39 public synchronized void reshape(int x, int y, int width, int height)
40 {
41     // dbg("_p_tb_db("+theTableNumber+").reshape("+x+", "+y+", "+width+",
42     "+height+")");
43
44     super.reshape(x, y, width, height);
45
46     if (!layoutComplete)
47         doLayout();
48
49     int iw = insets.left + insets.right;
50     int ih = insets.top + insets.bottom;
51
52     if (width > iw && height > ih
53         && (width != tableSize.width || height != tableSize.height))
54         this.reshapeAll(width, height);
55
56     repaint();
57 }
58
59 public final void reshapeAll()
60 {
61     // dbg("_p_tb_db("+theTableNumber+").reshapeAll()");
62
63     _p_tc_db firstCell = null;
64     for (int j = 0; j < nRows-1 && firstCell == null; j++)
65         for (int i = 0; i < nCols && firstCell == null; i++)
66             firstCell = tableCells[j][i];
67
68     _p_rs_db firstPanel = null;
69     if (firstCell != null && firstCell.countComponents() > 0)
70         if (firstCell.getComponent(0) instanceof _p_rs_db)
```

```

1      firstPanel = (_p_rs_db)firstCell.getComponent(0);
2
3      int sW =
4          this.size().width == 0
5          ?   Math.min
6              (   getToolkit().getScreenSize().width
7                  ,   (   firstPanel != null
8                      ?   firstPanel.standardLineWidth()
9                          :   180
10                     ) * nCols
11                  )
12          :   this.size().width
13          ;
14
15      int sH =
16          this.size().height == 0
17          ?   getToolkit().getScreenSize().height
18          :   this.size().height
19          ;
20
21      reshapeAll(sW, sH);
22  }
23
24  public final void reshapeAll(int sW, int sH)
25  {
26      // dbg("_p_tb_db"+"theTableNumber+").reshapeAll("+sW+", "+sH+");
27
28      int[] x = new int[nCols];   x[0] = insets.left;
29      int[] y = new int[nRows];   y[0] = insets.top;
30      int[] w = new int[nCols];
31      int[] h = new int[nRows];
32
33      boolean zoomableCells = true;           // optimistic, to be sure
34
35      Dimension p = prefTableSize();
36      Dimension m = minTableSize();
37
38      if (sW < m.width)                       // at least 1 col set below min width
39      {
40          zoomableCells = false;              // set false to zoom table instead
41
42          for (int i = 0; i < nCols-1; i++)
43          {
44              x[i+1] = Math.min
45                  (   sW - insets.right           // just truncate final column(s)
46                      ,   x[i] + minColWidth(i)
47                  );
48              w[i] = x[i+1] - x[i];            // result is always non-negative
49          }
50
51          w[nCols-1] = Math.max                // the last column
52              (   0
53                  ,   Math.min
54                      (   minColWidth(nCols-1)
55                          ,   sW - insets.right - x[nCols-1]
56                      )
57              );
58      }
59      else
60      if (sW < p.width)                       // at least 1 col set below pref
61      width
62      {
63          zoomableCells = false;              // set false to zoom table instead
64
65          int[] z = new int[nCols];
66          int nW = sW - insets.left - insets.right;
67
68          for (int i = 0; i < nCols; i++)
69          {
70              z[i] = minColWidth(i);

```



```

1      w[i] = prefColWidth(i);
2      nW -= w[i];
3  }
4      while (nW < 0)                                // decrement last column with most
5  slack
6      {
7          int xs = 0, xI = 0;
8          for (int i = 0; i < nCols; i++)
9              if (xs < w[i] - z[i])                // column slack = excess over minimum
10 width
11             xs = w[i] - z[xI = i];                // get the column number
12             w[xI]--;                                // narrow the target column
13             nW++;                                    // overshoot value increments closer
14 to zero
15         }
16         for (int i = 0; i < nCols-1; i++)
17             x[i+1] = x[i] + w[i];
18     }
19     else                                            // pref width is narrower than spec'd
20 width
21     {
22         for (int i = 0; i < nCols-1; i++)
23             x[i+1] = x[i] + (w[i] = prefColWidth(i));
24
25         w[nCols-1] = prefColWidth(nCols-1);
26     }
27
28     if (sH < m.height)                            // at least 1 row set below min
29 height
30     {
31         zoomableCells = false;                    // set false to zoom table instead
32
33         for (int j = 0; j < nRows-1; j++)
34         {
35             y[j+1] = Math.min
36             (   sH - insets.bottom,                // just truncate final row(s)
37               ,   y[j] + minRowHeight(j)
38             );
39             h[j] = y[j+1] - y[j];                    // result is always non-negative
40         }
41
42         h[nRows-1] = Math.max                        // the last row
43         (   0
44           ,   Math.min
45             (   minRowHeight(nRows-1)
46               ,   sH - insets.bottom - y[nRows-1]
47             )
48         );
49     }
50     else
51     if (sH < p.height)                            // at least 1 row set below pref
52 height
53     {
54         zoomableCells = false;                    // set false to zoom table instead
55
56         int[] z = new int[nRows];
57         int nH = sH - insets.top - insets.bottom;
58
59         for (int j = 0; j < nRows; j++)
60         {
61             z[j] = minRowHeight(j);
62             h[j] = prefRowHeight(j);
63             nH -= h[j];
64         }
65         while (nH < 0)                                // decrement last row with most slack
66         {
67             int xs = 0, xJ = 0;
68             for (int j = 0; j < nRows; j++)
69                 if (xs < h[j] - z[j])                // row slack = excess over minimum
70 height

```

```

1          xs = h[j] - z[xJ = j]; // get the row number
2          h[xJ]--; // narrow the target row
3          nH++; // overshoot value increments closer
4  to zero
5      }
6      for (int j = 0; j < nRows-1; j++)
7          y[j+1] = y[j] + h[j];
8      }
9      else // pref height is shorter than spec'd
10 height
11     {
12         for (int j = 0; j < nRows-1; j++)
13             y[j+1] = y[j] + (h[j] = prefRowHeight(j));
14
15         h[nRows-1] = prefRowHeight(nRows-1);
16     }
17
18     this.setZoomable(zoomableCells);
19
20     for (int j = 0; j < nRows; j++)
21     {
22         for (int i = 0; i < nCols; i++) if (tableCells[j][i] != null)
23         {
24             _p_tc_db c = tableCells[j][i];
25
26             int cWide = 0, cN = c.colNum;
27
28             for (int n = 0; n < c.colSpan && n < nCols - cN; n++)
29                 cWide += w[cN + n];
30
31             int cHigh = 0, rN = c.rowNum;
32             for (int n = 0; n < c.rowSpan && n < nRows - rN; n++)
33                 cHigh += h[rN + n];
34
35             // dbg("_p_tb_db("+theTableNumber+").reshape cell["+j+"]["+i+"]:
36             ("x[cN]+", "y[rN]+", "cWide+", "cHigh+");
37
38             c.reshape
39             (
40                 x[cN]
41                 , y[rN]
42                 , cWide
43                 , cHigh
44             );
45         }
46         tableSize = new Dimension(sW, sH);
47
48         // dbg("_p_tb_db("+theTableNumber+").reshapeAll("+sW+", "+sH+") complete: table
49         prefSize = "+p.width+"x"+p.height+"; miniSize = "+m.width+"x"+m.height);
50     }
51
52     public final void setFontSizeIndex(int newSizeIndex)
53     {
54         // dbg("_p_tb_db("+theTableNumber+").setFontSizeIndex("+newSizeIndex+"): # comps
55         = "+countComponents());
56
57         this.fontSizeIndex = newSizeIndex;
58         for (int i = 0; i < this.countComponents(); i++)
59         {
60             if (this.getComponent(i) instanceof _p_tc_db)
61             {
62                 ((_p_tc_db)this.getComponent(i)).
63                 setFontSizeIndex(fontSizeIndex);
64             }
65         }
66     }
67
68     public void setZoomable(boolean zoomableCells)
69     {

```

```

1      // dbg("_p_tb_db"+"theTableNumber+").setZoomable("+zoomableCells+"): # comps =
2      "+countComponents());
3
4      for (int i = 0; i < this.countComponents(); i++)
5      {
6          if (this.getComponent(i) instanceof _p_tc_db)
7          {
8              ((_p_tc_db)this.getComponent(i)).
9                  setZoomable(zoomableCells);
10         }
11     }
12 }
13
14 public synchronized void update(Graphics g)
15 {
16     // fillRect is NOT performed here to eliminate blank screen boredom during
17     offscreen drawing
18
19     // g.setColor(getBackground());
20     // g.fillRect(0, 0, width, height);
21     // g.setColor(getForeground());
22
23     paint(g);
24 }
25
26 public void validate()
27 {
28     // dbg("_p_tb_db"+"theTableNumber+").validate());
29
30     if (!this.isValid() && !layoutComplete)
31         doLayout();
32
33     super.validate();
34 }
35
36 public _p_tb_db(String tableSpec, Integer theTableNumber, ez_html ez_HTML)
37 {
38     // dbg("new _p_tb_db"+"tableSpec+", "theTableNumber+");
39
40     this.ez_HTML = ez_HTML;
41     this.tableSpec = tableSpec;
42     this.theTableNumber = theTableNumber;
43     this.setLayout(null);
44
45     insets.top      = vSpace + 1 + border + cellSpacing/2;
46     insets.left     = hSpace + 1 + border + cellSpacing/2;
47     insets.bottom   = vSpace + 1 + border + cellSpacing - cellSpacing/2;
48     insets.right    = hSpace + 1 + border + cellSpacing - cellSpacing/2;
49 }
50 }
51
52 /**
53  *  %W% %E% Everett Stoub
54  *
55  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
56  *
57  *  an object to store table cell data
58  *
59  *  this class is not downloaded unless needed
60  *
61  *  @author Everett Stoub %I%, %G%
62  */
63 import java.awt.*;
64 import java.util.*;
65
66 public class _p_tc_db extends Panel
67 {
68     public Color    cellBg      = null;
69     public Color    cellFg      = null;
70     public int      align       = _p_rs_db.LEFT;

```

```

1      public int      vAlign      = _p_rs_db.CENTER;
2      public int      minWidth    = 0;
3      public int      minHeight   = 0;
4      public int      colSpan     = 1;
5      public int      rowSpan     = 1;
6      public int      colNum      = 1;
7      public int      rowNum      = 1;
8      public int      border      = 0;
9      public int      cellPadding = 0;
10     public int      cellSpacing = 0;
11     public boolean   isData      = true;
12     public String    content     = _p_rs_db.MT;
13
14     private final void dbg(String s)
15     {
16         if (debug) System.out.println(s);
17     }
18     private boolean debug = false;          // Print debugging info?
19
20     public boolean mouseDown(Event evt, int x, int y)
21     {
22         // dbg("_p_tc_db.mouseDown("+x+", "+y+")");
23
24         return false;
25     }
26
27     public final void setFontSizeIndex(int newSizeIndex)
28     {
29         // dbg("_p_tc_db().setFontSizeIndex("+newSizeIndex+"): # comps =
30 "+countComponents());
31
32         for (int i = 0; i < this.countComponents(); i++)
33         {
34             if (this.getComponent(i) instanceof _p_rs_db)
35                 (( _p_rs_db)this.getComponent(i)).
36                 setFontSizeIndex(newSizeIndex);
37             else
38                 if (this.getComponent(i) instanceof _p_tb_db)
39                     (( _p_tb_db)this.getComponent(i)).
40                     setFontSizeIndex(newSizeIndex);
41         }
42     }
43
44     public void setZoomable(boolean zoomable)
45     {
46         // dbg("_p_tc_db().setZoomable("+zoomable+"): # comps = "+countComponents());
47
48         for (int i = 0; i < this.countComponents(); i++)
49         {
50             if (this.getComponent(i) instanceof _p_rs_db)
51             {
52                 (( _p_rs_db)this.getComponent(i)).
53                 setZoomable(zoomable);
54             }
55             else
56                 if (this.getComponent(i) instanceof _p_tb_db)
57                 {
58                     (( _p_tb_db)this.getComponent(i)).
59                     setZoomable(zoomable);
60                 }
61         }
62     }
63
64     public synchronized void update(Graphics g)
65     {
66         // fillRect is NOT performed here to eliminate blank screen boredom during
67         offscreen drawing
68
69         // g.setColor(getBackground());
70         // g.fillRect(0, 0, width, height);

```

```

1      // g.setColor(getForeground());
2
3      paint(g);
4  }
5
6  public _p_tc_db
7  (   Color   cellBg
8    ,   Color   cellFg
9    ,   int     align
10   ,   int     vAlign
11   ,   int     minWidth
12   ,   int     colSpan
13   ,   int     rowSpan
14   ,   int     minHeight
15   ,   int     border
16   ,   int     cellPadding
17   ,   int     cellSpacing
18   ,   boolean isData // false if header, true if data
19 )
20 {
21     this.cellBg       = cellBg;
22     this.cellFg       = cellFg;
23     this.align        = align;
24     this.vAlign       = vAlign;
25     this.minWidth     = minWidth;
26     this.colSpan      = colSpan;
27     this.rowSpan      = rowSpan;
28     this.minHeight    = minHeight;
29     this.border       = border;
30     this.cellPadding  = cellPadding;
31     this.cellSpacing  = cellSpacing;
32     this.isData       = isData;
33
34     this.setLayout(new BorderLayout());
35 }
36
37 /**
38  *  %W% %E% Everett Stoub
39  *
40  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
41  *
42  *  an object to handle html horizontal rules
43  *
44  *  this class is not downloaded unless needed
45  *
46  *
47  *  @author Everett Stoub %I%, %G%
48  */
49 import java.awt.*;
50 import java.io.*;
51 import java.net.*;
52
53 public class _o_hr_db extends Object
54 {
55     public static int placeRule
56     (   String tagText
57     ,   _p_rs_db panel
58     ,   int addIndexValue
59     ,   Graphics g
60     )
61     {
62         int addIndex = addIndexValue;
63         int itsSize = _o_tg_db.getTagAttribInt(tagText, panel.SZ, 3);
64         int itsWidth = _o_tg_db.getTagWidth(tagText, panel.effectiveWidth());
65
66         addIndex += panel.paintRow(g,0);
67         pending line elements
68         if (!panel.startNewParagraph(1))
69         {

```

// try to punch out

```

1      return -addIndex;                                // no more room on
2  the page
3      }
4      while (panel.yb + itsSize > panel.ye)
5      {
6          if (panel.lastColumn)
7          {
8              return -addIndex;                        // no more columns
9  left to fill
10         }
11         panel.nextColumn();
12     }
13     if (itsSize > 0 && itsWidth > 0)
14     {
15         int itsAlign = _o_tq_db.getTagAlignment(tagText, panel.CENTER);
16         int itsStart = panel.xb + panel.indent(false) +
17         itsAlign*(panel.effectiveWidth() - itsWidth)/2;
18
19         float[] hsbvals = new float[3];
20         hsbvals = Color.RGBtoHSB(panel.getBackground().getRed(),
21         panel.getBackground().getGreen(), panel.getBackground().getBlue(), hsbvals);
22         if (g != null)
23         {
24             if (hsbvals[2] > 0.5f)
25             {
26                 g.setColor(panel.getBackground().darker().darker());
27             }
28             else
29             {
30                 g.setColor(panel.getBackground().brighter().brighter());
31             }
32             if (tagText.indexOf("noshade") > -1)
33             {
34
35                 g.fillRoundRect(itsStart, panel.yb, itsWidth, itsSize, itsSize, itsSize);
36             }
37             else
38             {
39                 g.draw3DRect(itsStart, panel.yb, itsWidth, itsSize, false);
40             }
41         }
42         panel.yb += itsSize;
43         if (!panel.startNewParagraph(1))
44         {
45             return -addIndex;
46         }
47     }
48     // textAlign = LEFT;                                // it's supposed to be reset
49     this way, but PageMill does not!
50     return addIndex;
51 }
52 }
53
54 /**
55  *  %W% %E% Everett Stoub
56  *
57  *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
58  *
59  *  an object to handle html lists
60  *
61  *  this class is not downloaded unless needed
62  *
63  *  @author Everett Stoub %I%, %G%
64  */
65 import java.awt.*;
66 import java.io.*;
67 import java.net.*;
68
69 public class _o_li_db extends Object
70 {

```

```

1      public final static String ST          = "start";
2
3      // unordered list bullet types
4
5      public final static int    DISC        = 1;
6      public final static int    CIRCLE      = 2;
7      public final static int    BLOCK      = 3;
8      public final static int    SQUARE     = 4;
9
10     // ordered list numbering types
11
12     public final static int    UCROMAN     = 1;    // list type "I"
13     public final static int    UCLETTER    = 2;    // list type "A"
14     public final static int    ARABIC     = 3;    // list type "1"
15     public final static int    LCROMAN     = 4;    // list type "i"
16     public final static int    LCLETTER    = 5;    // list type "a"
17
18     public static int addItem
19     (   String tagText
20     ,   String itsText
21     ,   _p_rs_db panel
22     ,   int addIndexValue
23     ,   Graphics g
24     )
25     {
26         // dbg("_o_li_db.addItem(["+itsText+"],panel,"+addIndexValue+",g)");
27
28         int addIndex = addIndexValue;
29
30         if (panel.listAccount.size() > 0)
31         {
32             addIndex += panel.paintRow(g,0);           // try to punch out pending
33             line elements
34             if (!panel.startNewParagraph(0))
35             {
36                 return -addIndex;
37             }
38             if (tagText.startsWith(panel.LI))
39             {
40                 Point itsItem = (Point)panel.listAccount.firstElement();
41                 itsItem.x = _o_tg_db.getTagAttribInt
42                 (   tagText
43                 ,   ST
44                 ,   itsItem.x
45                 // optional start value
46                 // increment its counter in
47                 panel.paintRow
48                 );
49                 itsItem.y = _o_li_db.getTagAttribTypeInt
50                 (   itsText
51                 ,   itsItem.y
52                 // as asserted by user
53                 // inherited list element
54                 type
55                 );
56             }
57             else
58             {
59                 // a definition list element?
60                 panel.termDefinition
61                 = tagText.equals(panel.DD);
62                 // if true: added indentation
63                 panel.newListItem = true;
64             }
65             return addIndex;
66         }
67     }
68
69     public static int addList
70     (   String tagText
71     ,   String itsText
72     ,   _p_rs_db panel
73     ,   int addIndexValue
74     ,   Graphics g
75     )
76     {

```

```

1      int addIndex = addIndexValue;
2
3      addIndex += panel.paintRow(g,0);                // try to punch out pending
4  line elements
5      if (!panel.startNewParagraph((panel.listAccount.size() > 0)?0:1))
6      {
7          return -addIndex;
8      }
9      if (tagText.startsWith(panel.OL))
10     {
11         panel.orderedList = true;
12     }
13     else
14     {
15         panel.unorderList = true;
16     }
17     if (tagText.equals(panel.DL))
18     {
19         panel.listAccount.insertElementAt(new Point(1,0),0);
20     }
21     else
22     {
23         int litIndex = _o_tg_db.getTagAttribInt(tagText, ST, 1) - 1;
24         int listType = _o_li_db.getTagAttribTypeInt
25         (    itsText
26         ,    panel.listAccount.size() % (panel.unorderList?4:5) + 1
27         );
28
29         Point listElement =
30             new Point
31             (    litIndex                // start value
32             ,    listType                // element type
33             );
34         panel.listAccount.insertElementAt(listElement,0);
35     }
36
37     return addIndex;
38 }
39
40 public static int endList
41 {    String tagText
42 ,    String itsText
43 ,    _p_rs_db panel
44 ,    int addIndexValue
45 ,    Graphics g
46 }
47 {
48     int addIndex = addIndexValue;
49
50     addIndex += panel.paintRow(g,0);                // try to punch out
51 pending line elements
52     if (!panel.startNewParagraph((panel.listAccount.size() > 0)?0:1))
53     {
54         return -addIndex;
55     }
56     int itsSize = panel.listAccount.size();
57     if (itsSize == 1)
58     {
59         if (tagText.equals(panel.SL+panel.OL))
60         {
61             panel.orderedList = false;
62         }
63         else
64         {
65             panel.unorderList = false;
66         }
67     }
68     if (itsSize > 0)
69     {
70         panel.listAccount.removeElementAt(0);        // LIFO list

```



```

1      )
2
3      return addIndex;
4  }
5
6  private static final void dbg(String s)
7  {
8      if (debug) System.out.println(s);
9  }
10 private static boolean debug = false;    // Print debugging info?
11
12 public static void drawBullet
13 (    _p_rs_db panel
14     ,    int x
15     ,    int y
16     ,    Graphics g
17 )
18 {
19     if (panel.listAccount.size() == 0
20     || !(panel.listAccount.firstElement() instanceof Point)) return;
21
22     Point itsItem = (Point)panel.listAccount.firstElement();
23     int D = (panel.textHeight+3)/4;
24     if (g != null && itsItem.x > 0) switch (itsItem.y)
25     {
26         case DISC:      g.fillOval(x - 2*D - 3, y - D - 2, D + 1, D + 1);
27         case CIRCLE:    g.drawOval(x - 2*D - 3, y - D - 2, D + 1, D + 1);    break;
28         case BLOCK:     g.fillRect(x - 2*D - 1, y - D - 2, D, D);
29         case SQUARE:   g.drawRect(x - 2*D - 1, y - D - 2, D, D);            break;
30     }
31     panel.newListItem = false;
32 }
33
34 public static void drawLabel
35 (    _p_rs_db panel
36     ,    int x
37     ,    int y
38     ,    Graphics g
39 )
40 {
41     if (panel.listAccount.size() == 0
42     || !(panel.listAccount.firstElement() instanceof Point)) return;
43
44     Point itsItem = (Point)panel.listAccount.firstElement();
45     String itsLabel = ".";
46     if (itsItem.x > 0) switch (itsItem.y)
47     {
48         case UCROMAN:    // up to 100 entries before starting over
49         case LCROMAN:
50             {
51                 String lab = panel.MT;
52                 switch ((itsItem.x - 1) % 10 + 1)
53                 {
54                     case 1: lab = "i";                break;
55                     case 2: lab = "ii";               break;
56                     case 3: lab = "iii";              break;
57                     case 4: lab = "iv";               break;
58                     case 5: lab = "v";                break;
59                     case 6: lab = "vi";               break;
60                     case 7: lab = "vii";              break;
61                     case 8: lab = "viii";             break;
62                     case 9: lab = "ix";               break;
63                 }
64                 switch ((itsItem.x/10 - 1) % 10 + 1)
65                 {
66                     case 1: lab = "x"                + lab; break;
67                     case 2: lab = "xx"               + lab; break;
68                     case 3: lab = "xxx"              + lab; break;
69                     case 4: lab = "xl"                + lab; break;
70                     case 5: lab = "l"                 + lab; break;

```

```

1      case 6: lab = "lx"      + lab; break;
2      case 7: lab = "lxx"     + lab; break;
3      case 8: lab = "lxxx"    + lab; break;
4      case 9: lab = "xc"      + lab; break;
5      }
6      if (itsItem.y == UCROMAN)
7          lab = lab.toUpperCase();
8      itsLabel = lab + itsLabel;
9  }
10     break;
11     case UCLETTER: // up to 26**2 entries before starting over
12     case LCLETTER:
13     {
14         char[] lab = {(char)(97 + (itsItem.x - 1) % 26)};
15         itsLabel = (new String(lab)) + itsLabel;
16     }
17     if (itsItem.x > 26)
18     {
19         char[] lab = {(char)(97 + (itsItem.x/26 - 1) % 26)};
20         itsLabel = (new String(lab)) + itsLabel;
21     }
22     if (itsItem.y == UCLETTER)
23         itsLabel = itsLabel.toUpperCase();
24     break;
25     case ARABIC:
26         itsLabel = itsItem.x + itsLabel;
27     break;
28 }
29
30     if (g != null) g.drawString(itsLabel, x -
31 panel.textMetrics.stringWidth(itsLabel), y);
32     panel.newListItem = false;
33 }
34
35 public static final int getTagAttribTypeInt
36 ( String itsText
37 , int itsDefault
38 )
39 {
40     String itsLabel = _o_tg_db.getTagAttribString(itsText, _p_rs_db.TP);
41
42     if (itsLabel == null)
43     {
44         return itsDefault;
45     }
46
47     String lowLabel = itsLabel.toLowerCase();
48
49     if (lowLabel.equals("circle")) return CIRCLE;    else
50     if (lowLabel.equals("square")) return SQUARE;    else
51     if (lowLabel.equals("disc")) return DISC;        else
52
53     if (itsLabel.equals("1")) return ARABIC;         else
54     if (itsLabel.equals("A")) return UCLETTER;       else
55     if (itsLabel.equals("a")) return LCLETTER;       else
56     if (itsLabel.equals("I")) return UCROMAN;       else
57     if (itsLabel.equals("i")) return LCROMAN;       else
58
59     return itsDefault;
60 }
61 }
62
63 /**
64 *  %W% %E% Everett Stoub
65 *
66 *  Copyright (c) '96-'97 ION Web Products, Inc. All Rights Reserved.
67 *
68 *  platform-specific html character entity converter
69 *
70 *  Character Entity parsing honors the following characters

```

1	*							
2	*	NOTE: each JVM will yield some variations due to internal font implementations						
3	*							
4	*							
5	*	NAMED	NUMERIC	DESCRIPTION	ALL	MAC	ALT	
6	*	=====	=====	=====	==	==	==	
7	*	"	"	double quotation mark				
8	*	&	&	ampersand				
9	*	<	<	less than sign				
10	*	>	>	greater than sign				
11	*		‚	comma	44			
12	*		ƒ	florin		159	102	
13	*		„	right double quote		155	34	
14	*		…	ellipsis		138	"..."	
15	*		†	dagger		221	n/a	
16	*		‡	double dagger		253	n/a	
17	*		ˆ	circumflex		150	n/a	
18	*		‰	permil		148	"0/00"	
19	*		Š	underscore	95			
20	*		‹	less than sign	60			
21	*		Œ	capital OE ligature		145	"OE"	
22	*		‘	left single quote		140	39	
23	*		’	right single quote		140	39	
24	*		“	left double quote		155	34	
25	*		”	right double quote		155	34	
26	*		•	bullet		128	183	
27	*		–	em dash		173	45	
28	*		—	en dash		139	45	
29	*		˜	tilde	126			
30	*		™	trademark		129	"(TM)"	
31	*		š	underscore	95			
32	*		›	greater than sign	62			
33	*		œ	small OE ligature	"oe"			
34	*		Ÿ	capital Y umlaut		141	n/a	
35	*	 	 	nonbreaking space				
36	*	¡	¡	inverted exclamation point				
37	*	¢	¢	cents sign				
38	*	£	£	pound sign				
39	*	¤	¤	general currency sign				
40	*	¥	¥	Yen sign				
41	*	¦	¦	broken vertical bar		" "		
42	*	§	§	section mark				
43	*	¨	¨	umlaut				
44	*	©	©	copyright mark				
45	*	ª	ª	feminine ordinal				
46	*	«	«	left angle quote				
47	*	¬	¬	not sign				
48	*	­	­	soft hyphen				
49	*	®	®	registered trade mark				
50	*	¯	¯	macron accent				
51	*	°	°	degree sign				
52	*	±	±	plus or minus				
53	*	²	²	superscript 2		"2"		
54	*	³	³	superscript 3		"3"		
55	*	´	´	acute accent				
56	*	µ	µ	micro sign (Greek mu)				
57	*	¶	¶	paragraph sign				
58	*	·	·	middle dot mark				
59	*	¸	¸	cedilla				
60	*	¹	¹	superscript 1		"1"		
61	*	º	º	masculine ordinal				
62	*	»	»	right angle quote				
63	*	¼	¼	fraction one-fourth		"1/4"		
64	*	½	½	fraction one-half		"1/2"		
65	*	¾	¾	fraction three-fourths		"3/4"		
66	*	¿	¿	inverted question mark				
67	*	À	À	cap A with grave accent				
68	*	Á	Á	cap A with acute accent				
69	*	Â	Â	cap A with circumflex accent				
70	*	Ã	Ã	cap A with tilde accent				

```

1      *      &Auml;      &#196; cap A with unlaut accent
2      *      &Aring;     &#197; cap A with ring accent
3      *      &AElig;     &#198; cap AE ligature
4      *      &Ccedil;    &#199; cap cedilla
5      *      &Egrave;    &#200; cap E with grave accent
6      *      &Eacute;    &#201; cap E with acute accent
7      *      &Ecirc;     &#202; cap E with circumflex accent
8      *      &Euml;      &#203; cap E with unlaut accent
9      *      &Igrave;    &#204; cap I with grave accent
10     *      &Iacute;    &#205; cap I with acute accent
11     *      &Icirc;     &#206; cap I with circumflex accent
12     *      &Iuml;      &#207; cap I with unlaut accent
13     *      &ETH;       &#208; cap eth, Icelandic      127
14     *      &Ntilde;    &#209; cap N with tilde accent
15     *      &Ograve;    &#210; cap O with grave accent
16     *      &Oacute;    &#211; cap O with acute accent
17     *      &Ocirc;     &#212; cap O with circumflex accent
18     *      &Otilde;    &#213; cap O with tilde accent
19     *      &Ouml;      &#214; cap O with unlaut accent
20     *      &times;     &#215; multiply sign      183
21     *      &Oslash;    &#216; cap O with slash
22     *      &Ugrave;    &#217; cap U with grave accent
23     *      &Uacute;    &#218; cap U with acute accent
24     *      &Ucirc;     &#219; cap U with circumflex accent
25     *      &Uuml;      &#220; cap U with unlaut accent
26     *      &Yacute;    &#221; cap Y with acute accent      89
27     *      &THORN;     &#222; cap THORN, Icelandic      127
28     *      &szlig;     &#223; small sz ligature, German
29     *      &agrave;     &#224; small A with grave accent
30     *      &aacute;     &#225; small A with acute accent
31     *      &acirc;     &#226; small A with circumflex accent
32     *      &atilde;    &#227; small A with tilde accent
33     *      &auml;      &#228; small A with unlaut accent
34     *      &aring;     &#229; small A with ring accent
35     *      &aelig;     &#230; small AE ligature
36     *      &ccedil;    &#231; small cedilla
37     *      &egrave;    &#232; small E with grave accent
38     *      &eacute;    &#233; small E with acute accent
39     *      &ecirc;     &#234; small E with circumflex accent
40     *      &euml;      &#235; small E with unlaut accent
41     *      &igrave;    &#236; small I with grave accent
42     *      &iacute;    &#237; small I with acute accent
43     *      &icirc;     &#238; small I with circumflex accent
44     *      &iuml;      &#239; small I with unlaut accent
45     *      &eth;       &#240; small eth, Icelandic      143
46     *      &ntilde;    &#241; small N with tilde accent
47     *      &ograve;    &#242; small O with grave accent
48     *      &oacute;    &#243; small O with acute accent
49     *      &ocirc;     &#244; small O with circumflex accent
50     *      &otilde;    &#245; small O with tilde accent
51     *      &ouml;      &#246; small O with unlaut accent
52     *      &divide;    &#247; divide sign
53     *      &oslash;    &#248; small O with slash
54     *      &ugrave;    &#249; small U with grave accent
55     *      &uacute;    &#250; small U with acute accent
56     *      &ucirc;     &#251; small U with circumflex accent
57     *      &uuml;      &#252; small U with unlaut accent
58     *      &yacute;    &#253; small Y with acute accent      121
59     *      &ythorn;    &#254; small THORN, Icelandic      127
60     *      &yuml;      &#255; small Y with unlaut accent
61     *
62     *
63     * @author Everett Stoub %I%, %G%
64     */
65     import java.awt.*;
66     import java.util.*;
67
68     public class _o_nt_db extends Object
69     {
70         public String convertCharacterEntities(String itsText)

```

```

1      {
2          //  dbg("_o_nt_db.convertCharacterEntities(["+itsText+"]");
3
4          if (itsText == null)
5              return itsText;
6
7          boolean mac = System.getProperty("os.name").equals("macos");
8
9          StringTokenizer entities = new StringTokenizer(itsText, "&", true);
10         StringBuffer newText = new StringBuffer(70);
11
12         while (entities.hasMoreTokens())
13         {
14             String entity =                                //  [body text], [&][#nnn;body
15 text],..., [&][#nnn;body text]
16                 entities.nextToken("&");
17
18             if (entity.equals("&")
19                 && entities.hasMoreTokens())
20             {
21                 String eText =                                //  should be character entity
22 text #nnn or name
23                     entities.nextToken(";");
24
25                 if (entities.hasMoreTokens())
26                     entity = entities.nextToken(); //  should be ";" if eText is
27 valid character entity value
28                 else
29                     entity = _p_rs_db.MT;                //  avoid making things up
30
31                 if (entity.equals(";"))                    //  replace character entity with
32 local character (set)
33                 {
34                     int e = 0;
35                     if (!eText.startsWith("#"))            //  named entity: assign numeric
36 entity value
37                     {
38                         if (eText.equals("quot")) e = 34;   else
39                         if (eText.equals("amp")) e = 38;   else
40                         if (eText.equals("lt")) e = 60;    else
41                         if (eText.equals("gt")) e = 62;    else
42                         if (eText.equals("nbsp")) e = 160; else
43                         if (eText.equals("iexcl")) e = 161; else
44                         if (eText.equals("cent")) e = 162; else
45                         if (eText.equals("pound")) e = 163; else
46                         if (eText.equals("curren")) e = 164; else
47                         if (eText.equals("yen")) e = 165;  else
48                         if (eText.equals("brvbar")) e = 166; else
49                         if (eText.equals("sect")) e = 167; else
50                         if (eText.equals("uml")) e = 168;  else
51                         if (eText.equals("copy")) e = 169; else
52                         if (eText.equals("ordf")) e = 170; else
53                         if (eText.equals("laguo")) e = 171; else
54                         if (eText.equals("not")) e = 172;  else
55                         if (eText.equals("shy")) e = 173;  else
56                         if (eText.equals("reg")) e = 174;  else
57                         if (eText.equals("macr")) e = 175;  else
58                         if (eText.equals("deg")) e = 176;  else
59                         if (eText.equals("plusmn")) e = 177; else
60                         if (eText.equals("sup2")) e = 178; else
61                         if (eText.equals("sup3")) e = 179; else
62                         if (eText.equals("acute")) e = 180; else
63                         if (eText.equals("micro")) e = 181; else
64                         if (eText.equals("para")) e = 182; else
65                         if (eText.equals("middot")) e = 183; else
66                         if (eText.equals("cedil")) e = 184; else
67                         if (eText.equals("sup1")) e = 185;  else
68                         if (eText.equals("ordm")) e = 186;  else
69                         if (eText.equals("raquo")) e = 187;  else
70                         if (eText.equals("frac14")) e = 188;  else

```

```

1      if (eText.equals("frac12")) e = 189;      else
2      if (eText.equals("frac34")) e = 190;      else
3      if (eText.equals("iquest")) e = 191;      else
4      if (eText.equals("Agrave")) e = 192;      else
5      if (eText.equals("Aacute")) e = 193;      else
6      if (eText.equals("Acirc")) e = 194;      else
7      if (eText.equals("Atilde")) e = 195;      else
8      if (eText.equals("Auml")) e = 196;      else
9      if (eText.equals("Aring")) e = 197;      else
10     if (eText.equals("AElig")) e = 198;      else
11     if (eText.equals("Ccedil")) e = 199;      else
12     if (eText.equals("Egrave")) e = 200;      else
13     if (eText.equals("Eacute")) e = 201;      else
14     if (eText.equals("Ecirc")) e = 202;      else
15     if (eText.equals("Euml")) e = 203;      else
16     if (eText.equals("Igrave")) e = 204;      else
17     if (eText.equals("Iacute")) e = 205;      else
18     if (eText.equals("Icirc")) e = 206;      else
19     if (eText.equals("Iuml")) e = 207;      else
20     if (eText.equals("ETH")) e = 208;      else
21     if (eText.equals("Ntilde")) e = 209;      else
22     if (eText.equals("Ograve")) e = 210;      else
23     if (eText.equals("Oacute")) e = 211;      else
24     if (eText.equals("Ocirc")) e = 212;      else
25     if (eText.equals("Otilde")) e = 213;      else
26     if (eText.equals("Ouml")) e = 214;      else
27     if (eText.equals("times")) e = 215;      else
28     if (eText.equals("Oslash")) e = 216;      else
29     if (eText.equals("Ugrave")) e = 217;      else
30     if (eText.equals("Uacute")) e = 218;      else
31     if (eText.equals("Ucirc")) e = 219;      else
32     if (eText.equals("Uuml")) e = 220;      else
33     if (eText.equals("Yacute")) e = 221;      else
34     if (eText.equals("THORN")) e = 222;      else
35     if (eText.equals("szlig")) e = 223;      else
36     if (eText.equals("agrave")) e = 224;      else
37     if (eText.equals("aacute")) e = 225;      else
38     if (eText.equals("acirc")) e = 226;      else
39     if (eText.equals("atilde")) e = 227;      else
40     if (eText.equals("auml")) e = 228;      else
41     if (eText.equals("aring")) e = 229;      else
42     if (eText.equals("aelig")) e = 230;      else
43     if (eText.equals("ccedil")) e = 231;      else
44     if (eText.equals("egrave")) e = 232;      else
45     if (eText.equals("eacute")) e = 233;      else
46     if (eText.equals("ecirc")) e = 234;      else
47     if (eText.equals("euml")) e = 235;      else
48     if (eText.equals("igrave")) e = 236;      else
49     if (eText.equals("iacute")) e = 237;      else
50     if (eText.equals("acirc")) e = 238;      else
51     if (eText.equals("iuml")) e = 239;      else
52     if (eText.equals("eth")) e = 240;      else
53     if (eText.equals("ntilde")) e = 241;      else
54     if (eText.equals("ograve")) e = 242;      else
55     if (eText.equals("oacute")) e = 243;      else
56     if (eText.equals("ocirc")) e = 244;      else
57     if (eText.equals("otilde")) e = 245;      else
58     if (eText.equals("ouml")) e = 246;      else
59     if (eText.equals("divide")) e = 247;      else
60     if (eText.equals("oslash")) e = 248;      else
61     if (eText.equals("ugrave")) e = 249;      else
62     if (eText.equals("uacute")) e = 250;      else
63     if (eText.equals("ucirc")) e = 251;      else
64     if (eText.equals("uuml")) e = 252;      else
65     if (eText.equals("yacute")) e = 253;      else
66     if (eText.equals("yuml")) e = 255;      else
67     {
68         e = 32;      // space
69     }

```

```
1          // dbg("_o_nt_db.convertCharacterEntities(): the named character
2 entity &"+eText+"; was mapped to numeric code "+e);
3      }
4      else                                     // numeric entity: look after the
5 pound sign
6      {
7          try
8          {
9              e = Integer.parseInt(eText.substring(1));
10         }
11         catch (NumberFormatException ne)
12         {
13             // dbg("_o_nt_db.convertCharacterEntities(): the numeric
14 entity &"+eText+"; was not parsed");
15             e = 160;                          // nonbreaking space
16         };
17     }
18
19     // character or string remapping (internal font mapping exceptions)
20
21     String g = null;
22     switch (e)
23     {
24         case 130: e = 44;          break;
25         case 138:
26         case 154: e = 95;          break;
27         case 139: e = 60;          break;
28         case 152: e = 126;         break;
29         case 155: e = 62;          break;
30         case 156: g = "oe";        break;
31     }
32     if (mac) switch (e)
33     {
34         case 131: e = 159;         break;
35         case 132:
36         case 147:
37         case 148: e = 155;         break;
38         case 133: e = 138;         break;
39         case 134: e = 221;         break;
40         case 135: e = 253;         break;
41         case 136: e = 150;         break;
42         case 137: e = 148;         break;
43         case 140: e = 145;         break;
44         case 145:
45         case 146: e = 140;         break;
46         case 149: e = 128;         break;
47         case 150: e = 173;         break;
48         case 151: e = 139;         break;
49         case 153: e = 129;         break;
50         case 159: e = 141;         break;
51         case 166: g = "|";         break;
52         case 178: g = "2";         break;
53         case 179: g = "3";         break;
54         case 185: g = "1";         break;
55         case 188: g = "1/4";       break;
56         case 189: g = "1/2";       break;
57         case 190: g = "3/4";       break;
58         case 208:
59         case 222:
60         case 254: e = 127;         break;
61         case 215: e = 183;         break;
62         case 221: e = 89;          break;
63         case 240: e = 143;         break;
64         case 253: e = 121;         break;
65     }
66     else switch (e)
67     {
68         case 131: e = 102;         break;
69         case 132:
70         case 147:
```

```

1      case 148: e = 34;      break;
2      case 133: g = "...";  break;
3      case 137: g = "0/00"; break;
4      case 140: g = "OE";   break;
5      case 145:
6      case 146: e = 39;      break;
7      case 149: e = 183;    break;
8      case 150:
9      case 151: e = 45;      break;
10     case 153: g = "(TM)";  break;
11
12     }
13     char[] sub = {(char)e};
14     newText.append((g == null? new String(sub):g));
15
16     //  dbg("_o_nt_db.convertCharacterEntities(): the character entity
17     &" + eText + "; converted to [" + (g == null? new String(sub): g) + "]" );
18     }
19     else // syntax error:
20     {
21         newText.append("&").append(eText).append(entity);
22     }
23     else
24     {
25         newText.append(entity); // standard text
26     }
27     }
28     return newText.toString();
29 }
30
31 public static final void dbg(String s)
32 {
33     if (debug) System.out.println(s);
34 }
35 private static boolean debug = false; // Print debugging info?
36 }
37
38 /**
39  * sample web page with applet tag
40  */
41 <HTML>
42 <HEAD>
43     <META NAME="GENERATOR" CONTENT="Adobe PageMill 2.0 Mac">
44     <TITLE></TITLE>
45 </HEAD>
46 <BODY>
47
48 <APPLET CODE="ez_html.class" CODEBASE="ez_h_cls/" WIDTH="640" HEIGHT="480"
49 ALIGN="BOTTOM">
50 <PARAM NAME="fileName" VALUE="ezsample.htm">
51 <PARAM NAME="maxColumns" VALUE="4">
52 </APPLET>
53
54 </BODY>
55 </HTML>
56
57 /**
58  * ezsample.htm web page for applet
59  */
60 <HTML>
61 <HEAD>
62     <META NAME="GENERATOR" CONTENT="Adobe PageMill 2.0 Mac">
63     <TITLE>EZ HTML Sample Article</TITLE>
64 </HEAD>
65 <BODY BGCOLOR="#e8e8e8">
66
67 <P>EZ HTML is an <FONT COLOR="#00BA00">applet</FONT> for delivering substantial
68 text and graphics content while enhancing your web page's appearance. A
69 mouse click on the applet will evoke a reader screen window, or if over
70 a hypertext link, will open the <A HREF="ez_html.html">selected page with

```



```
1 a really long hypertext label which can show up in multiple lines and across
2 column and page boundaries</A> in the browser.</P>
3
4 <P>NOTE: this is a test document, in which samples of various html features
5 are rendered by the EZ HTML applet for comparison with standard browser
6 rendering. The table cells below contains very little information (most
7 or all of each cell should be displayed). Any cell can be viewed in a reading
8 window by clicking in it. The status bar provides context-sensitive help
9 text. You can also click on a table or cell boundary to view the entire
10 table in a larger window. EZ HTML is still in alpha - not all features have
11 been implemented, and not all implemented features are fully debugged.</P>
12
13 <P><TABLE BORDER="1" CELSPACING="0" CELLPADDING="5">
14 <CAPTION ALIGN="BOTTOM">Kumquat versus a poked eye, by gender</CAPTION>
15 <TR>
16 <TD COLSPAN="2" ROWSPAN="2"><FONT COLOR="#0000FF">Note: This table is taken from
17 <I>HTML,
18 The Definitive Guide,</I> by Chuck Musciano and Bill Kennedy, an O'Reilly
19 & Associates, Inc. publication. The table is merely designed to illustrate
20 multiple-row and column headers, blank areas and other special aspects to
21 HTML table structures.</FONT></TD>
22 <TH COLSPAN="2"><I><FONT COLOR="#FF0000" SIZE=+2>Preference</FONT></I></TH></TR>
23 <TR>
24 <TH>Eating Kumquats</TH>
25 <TH>Poke In The Eye</TH></TR>
26 <TR ALIGN="CENTER">
27 <TH ROWSPAN="2">Gender</TH>
28 <TH>Male</TH>
29 <TD>73%</TD>
30 <TD>27%</TD></TR>
31 <TR ALIGN="CENTER">
32 <TH>Female</TH>
33 <TD>16%</TD>
34 <TD>84%</TD></TR>
35 </TABLE>
36 </P>
37
38 <P ALIGN="CENTER"><FONT COLOR="#330099" SIZE=+1><HR NOSHADE>&quot;Click on
39 the applet&quot;;<HR NOSHADE></FONT></P>
40
41 <P>The reader screen presentation of content is interactive and optimized
42 for readability. Font size can be changed by pressing up or down arrow keys,
43 or by clicking the Font+ or Font- buttons. Pages can be turned by pressing
44 home, left, or right keys, or by clicking the Home, Previous, or Next buttons.</P>
45
46 <P ALIGN="CENTER"><FONT COLOR="#330099" SIZE=+1><HR NOSHADE>&quot;optimized
47 for readability&quot;;<HR NOSHADE></FONT></P>
48
49 <P>One of the key advantages of this reader screen presentation is its careful
50 avoidance of a particular form of eye-strain known as optokinetic nystagmus.
51 Nystagmus, a reflexive jump in the muscles directing the eye, is triggered
52 by, for example, vertical scrolling of text on a computer screen.</P>
53
54 <P ALIGN="CENTER"><IMG SRC="ez_eye.gif" WIDTH="74" HEIGHT="72" NATURALSIZEFLAG=
55 "3" ALIGN="BOTTOM"></P>
56
57 <P>As nystagmus strain increases, muscle jumps can become exaggerated and
58 even erratic, finally resulting in a perception of visual weariness. Nystagmus
59 is well-known as a side-effect of excessive alcohol: a simple road-side
60 test conducted by law officers can aid in determination of a possible DWI
61 offense. Other possible causes of nystagmus include general physical exhaustion
62 and illness.</P>
63
64 <P>While the term &quot;nystagmus&quot; may have been unfamiliar, the condition
65 is encountered in many circumstances, none of which is very pleasant.</P>
66
67 <P ALIGN="CENTER"><FONT COLOR="#330099" SIZE=+1><HR NOSHADE>&quot;optokinetic
68 nystagmus&quot;;<HR NOSHADE></FONT></P>
69
70 <P>The main side effect of optokinetic nystagmus induced by scrolling text
```

```
1 is a tendency to print the document, so that it can be read later with greater
2 comfort. This interrupts delivery of content and consumes resources. It
3 is very possible that the demise of the paperless office dreams of the '80's
4 was caused in part by nystagnus. Another major cause of its demise was the
5 difference in formatting between standard paper, height being typically
6 30% greater than width, and standard computer monitors, width being typically
7 33% greater than height. Word processors targeted paper formatting, so that
8 documents delivered from computer to computer were not easily adapted to
9 screen formats. This pattern also resulted in a tendency to print such documents
10 rather than struggle to consume their contents on screen.</P>
11
12 <P>EZ HTML solves these twin problems nicely. First, there is no scrolling
13 text, resulting in reduced eye-strain. Font sizing can be adjusted by the
14 reader for his particular situation of lighting, monitor quality and size,
15 and his personal visual comfort. Second, the content is automatically laid
16 out in columns of text, with neither too few words in a line, about 30 characters
17 minimum, nor too many, about 60 characters maximum. The upper limit yields
18 a line length which, as the human eye executes its rapid raster motion to
19 the beginning of the next line, is not too long to accurately and easily
20 find the correct next new line. If lines are much shorter, below the minimum,
21 lines of text contain too little content, and the frequent raster motions
22 can induce eye-strain.</P>
23
24 <P ALIGN=CENTER><FONT COLOR="#330099" SIZE=+1><HR NOSHADE>&quot;Web pages
25 ... become a rewarding experience with EZ&nbsp;HTML.&quot;<HR NOSHADE></FONT></P>
26
27 <P>This sample document contains about 3300 characters in about 640 words.
28 At normal reading speeds, it should take about 5 minutes to absorb its main
29 content. The effort required to read this article on the screen, using EZ
30 HTML, should be noticeably less than required by other channels. For instance,
31 using a word processor, two approaches are common. The first is to print
32 it out, collect the page(s), and finally read it. Alternatively, one could
33 open the document in a word processor, adjust window widths and zoom factors,
34 and finally read it, occasionally scrolling to bring unseen parts into view.
35 With EZ HTML, just sit back, tap the up arrow to adjust the font, and tap
36 the right arrow to turn the page as needed. Web pages, known widely as challenging
37 when it comes to delivering on-screen content, can become a rewarding experience
38 with EZ HTML.</P>
39
40 <P><IMG SRC="iwprmdlgo.gif" WIDTH="186" HEIGHT="181" ALIGN="BOTTOM" NATURALSIZEFLAG=
41 "3"></P>
42
43 <P ALIGN=CENTER><IMG SRC="otsdetails.gif" WIDTH="630" HEIGHT="509" NATURALSIZEFLAG=
44 "3" ALIGN="BOTTOM"></P>
45
46 <OL>
47   <LI>numbered list 1 level 1 item 1
48   <OL>
49     <LI>numbered list 2 level 2 item 1
50     <LI>numbered list 2 level 2 item 2
51     <LI>numbered list 2 level 2 item 3
52   </OL>
53   <LI>numbered list 1 level 1 item 2
54   <OL>
55     <LI>numbered list 3 level 2 item 1
56     <LI>numbered list 3 level 2 item 2
57   </OL>
58     <LI>numbered list 4 level 3 item 1
59     <OL>
60       <LI>numbered list 5 level 4 item 1
61     </OL>
62       <LI>numbered list 6 level 5 item 1
63       <LI>numbered list 6 level 5 item 2
64       <LI>numbered list 6 level 5 item 3
65     </OL>
66     <LI>numbered list 5 level 4 item 2
67     <LI>numbered list 5 level 4 item 3
68   </OL>
69   <LI>numbered list 4 level 3 item 2
70   <LI>numbered list 4 level 3 item 3
```

```

1      </OL>
2      <LI>numbered list 3 level 2 item 3
3      <LI>numbered list 3 level 2 item 4
4      </OL>
5      <LI>numbered list 1 level 1 item 3
6  </OL>
7
8  <DL>
9      <DT>term 1
10     <DD>definition 1 the wordy part of the two part definition item
11     <DT>term 2
12     <DD>definition 2 the wordy part of the two part definition item
13 </DL>
14
15 <UL>
16 <LI>bullet list 1 level 1 item 1
17 <UL>
18 <LI>bullet list 2 level 2 item 1
19 <UL>
20 <LI>bullet list 3 level 3 item 1
21 <UL>
22 <LI>bullet list 4 level 4 item 1
23 <LI>bullet list 4 level 4 item 2
24 <UL>
25 <LI>bullet list 5 level 5 item 1
26 <LI>bullet list 5 level 5 item 2
27 <LI>bullet list 5 level 5 item 3
28 </UL>
29 <LI>bullet list 4 level 4 item 3
30 </UL>
31 <LI>bullet list 4 level 3 item 2
32 <LI>bullet list 4 level 3 item 3
33 </UL>
34 <LI>bullet list 2 level 2 item 2
35 <LI>bullet list 2 level 2 item 3
36 </UL>
37 <LI>bullet list 1 level 1 item 2
38 <UL>
39 <LI>bullet list 6 level 2 item 1
40 <LI>bullet list 6 level 2 item 2
41 <UL>
42 <LI>bullet list 7 level 3 item 1
43 <LI>bullet list 7 level 3 item 2
44 <LI>bullet list 7 level 3 item 3
45 </UL>
46 <LI>bullet list 6 level 2 item 3
47 </UL>
48 <LI>bullet list 1 level 1 item 3
49 </UL>
50
51 <P>This is normal paragraph text. <B>This is bold text.</B> This is normal
52 paragraph text. <FONT SIZE=+1>This is big text.</FONT> This is normal paragraph
53 text. <CITE>This is a citation.</CITE> This is normal paragraph text. <CODE>This
54 is code text.</CODE> This is normal paragraph text. <EM>This is emphasis
55 text.</EM> This is normal paragraph text.</P>
56
57 <BLOCKQUOTE>
58 <P>This is blockquote justified text. <I>This is italic text.</I> This
59 is normal paragraph text. <KBD>This is keyboard text.</KBD> This is normal
60 paragraph text. <FONT SIZE=-1>This is small text.</FONT> This is normal
61 paragraph text <strike>This is strike text.</strike> This is normal paragraph text.
62 <STRONG>This
63 is strong text.</STRONG> This is normal paragraph text. <sub>This is subscript
64 text.</sub> This is normal paragraph text. <sup>This is superscript text.</sup>
65 This is normal paragraph text.</P>
66 </BLOCKQUOTE>
67
68 <P>This is left justified paragraph text. <TT>This is teletype text.</TT>
69 This is normal paragraph text. <u>This is underline text.</u> This is normal
70 paragraph text. <VAR>This is variable text.</VAR> This is normal paragraph

```

12/8/97

EZ HTML Source Code

p. 111

```
1 text.</P>
2
3 </BODY>
4 </HTML>
5
```